

计 算 机 科 学 丛 书



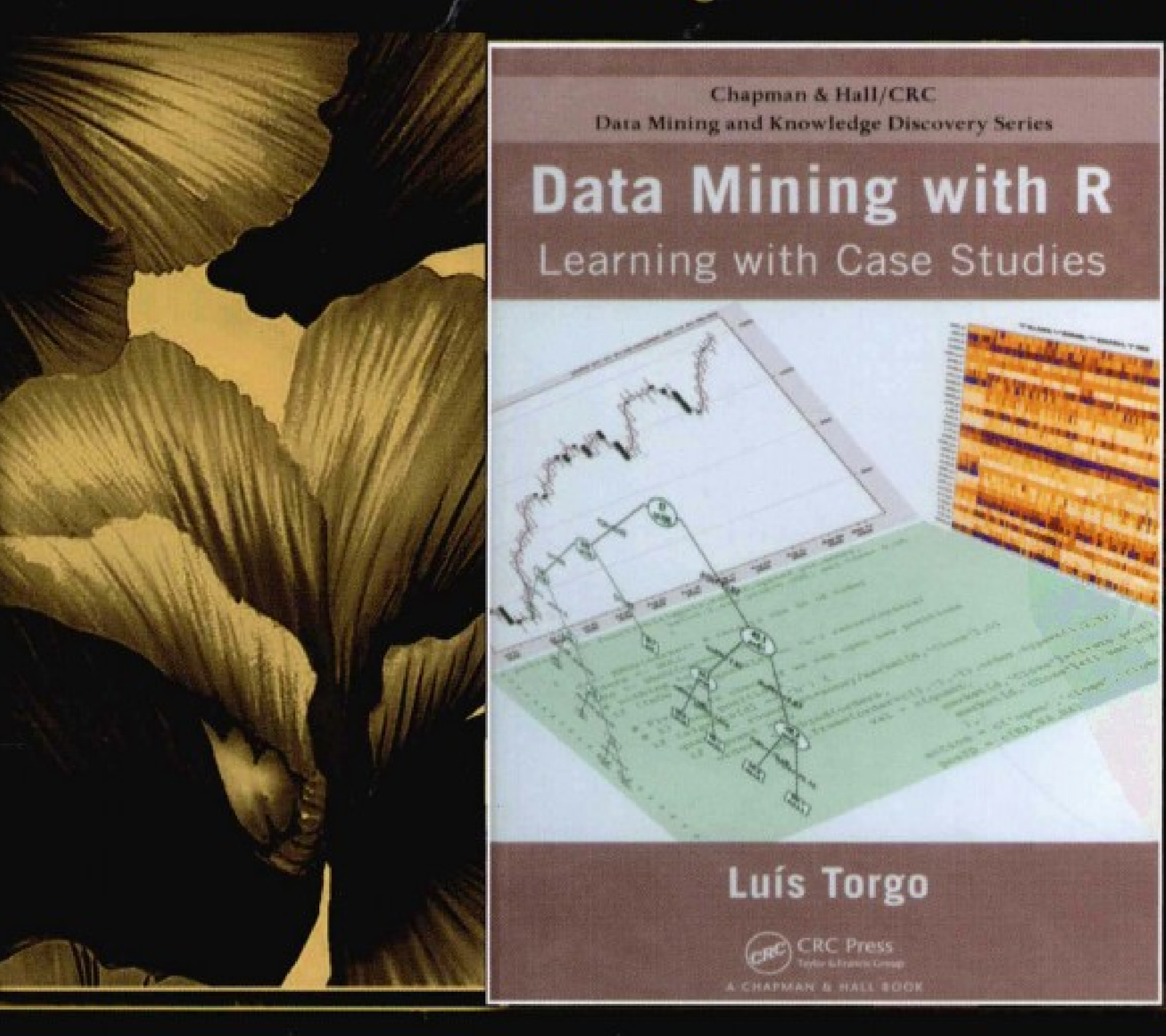
数据挖掘与R语言

（葡）Luís Torgo 著

李洪成 陈道轮 吴立明 译

Data Mining with R

Learning with Case Studies







机械工业出版社
China Machine Press

数据挖掘与R语言

Data Mining with R Learning with Case Studies

“如果你想学习如何用一款统计专家和数据挖掘专家所开发的免费软件包，那就选这本书吧。本书包括大量实际案例，它们充分体现了R软件的广度和深度。”

—— **Bernhard Pfahringer**，新西兰怀卡托大学

本书利用大量给出必要步骤、代码和数据的具体案例，详细描述了数据挖掘的主要过程和技术，广泛涵盖数据大小、数据类型、分析目标、分析工具等方面的各种具有挑战性的问题。

本书的支持网站（http://www.liaad.up.pt/~ltorgo/DataMiningWithR/）给出了案例研究的所有代码、数据集以及R函数包。

本书特色

- 通过仔细选择的案例涵盖了主要的数据挖掘技术。
- 给出的代码和方法可以方便地复制或者改编后应用于自己的问题。
- 不要求读者具有R、数据挖掘或统计技术的基础知识。
- 包含R和MySQL基础知识的简介。
- 提供了对数据挖掘技术的特性、缺点和分析目标的基本理解。

作者简介



Luís Torgo 葡萄牙波尔图大学计算机科学系副教授，现在在LIAAD实验室从事研究工作。他是APPIA会员，同时还是OBEGEF的创办会员。





客服热线：(010) 88378991 88361066
购书热线：(010) 68326294 88379649 68995259
投稿热线：(010) 88379604

读者信箱：hzjsj@hzbook.com
华章网站：www.hzbook.com
网上购书：www.china-pub.com

上架指导：计算机/数据挖掘



定价：49.00元

计 算 机 科 学 丛 书

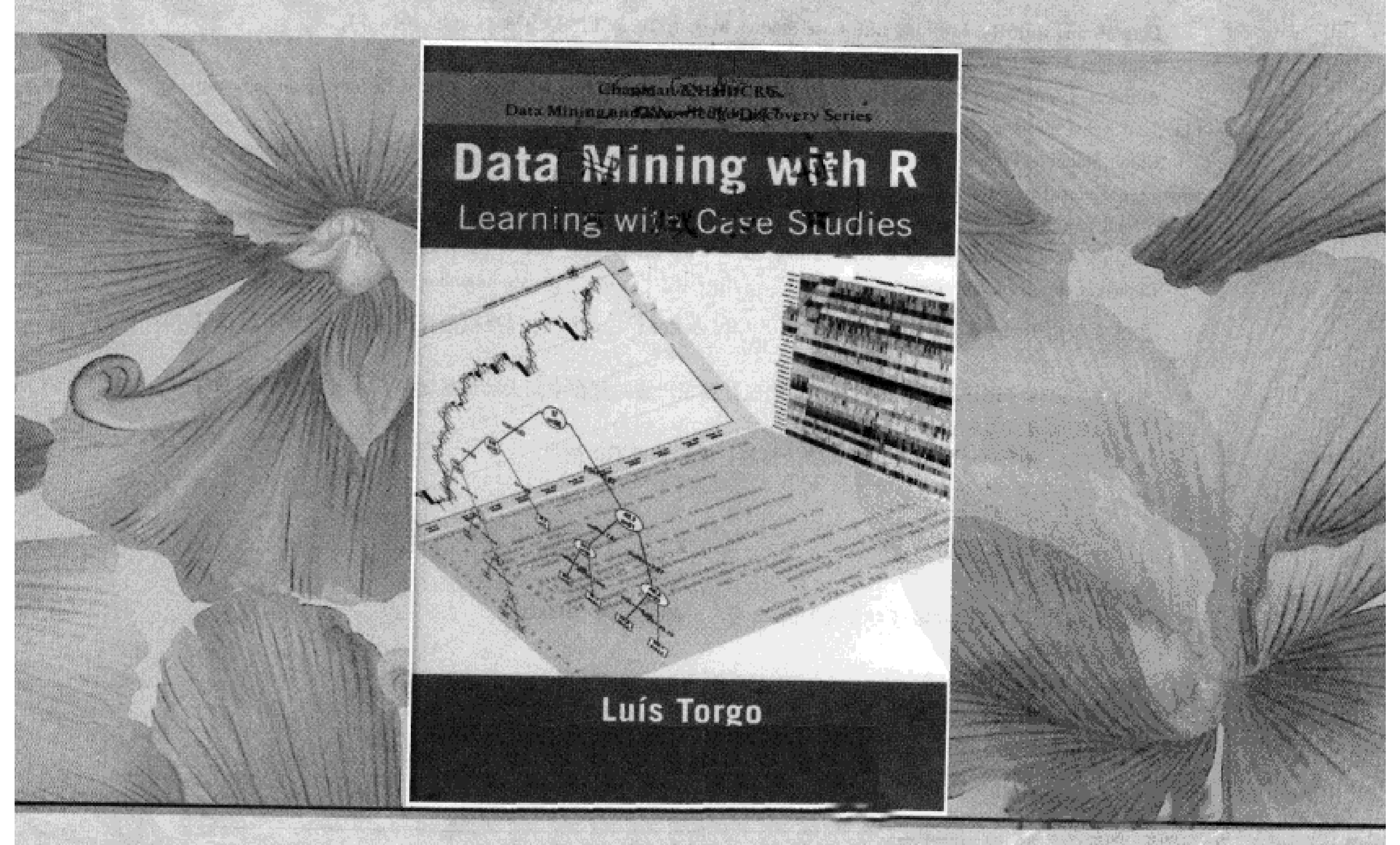
数据挖掘与R语言

（葡）Luís Torgo 著

李洪成 陈道轮 吴立明 译

Data Mining with R

Learning with Case Studies



机械工业出版社
China Machine Press

图书在版编目（CIP）数据

数据挖掘与R语言/（葡）托尔戈（Torgo，L.）著；李洪成等译．—北京：机械工业出版社，2013.2
（计算机科学丛书）
书名原文：Data Mining with R：Learning with Case Studies

ISBN 978-7-111-40700-3

Ⅰ．数…　Ⅱ．①托…　②李…　Ⅲ．①数据采集　②程序语言－程序设计　Ⅳ．①TP274　②TP312

中国版本图书馆CIP数据核字（2012）第302931号

本书版权登记号：图字：01-2012-0226

本书首先简要介绍了R软件的基础知识（安装、R数据结构、R编程、R的输入和输出等）。然后通过四个数据挖掘的实际案例（藻类频率的预测、证券趋势预测和交易系统仿真、交易欺诈预测、微阵列数据分类）介绍数据挖掘技术。这四个案例基本覆盖了常见的数据挖掘技术，从无监督的数据挖掘技术、有监督的数据挖掘技术到半监督的数据挖掘技术。全书以实际问题、解决方案和对解决方案的讨论为主线来组织内容，脉络清晰，并且各章自成体系。读者可以从头至尾逐章学习，也可以根据自己的需要进行学习，找到自己实际问题的解决方案。

本书不需要读者具备R和数据挖掘的基础知识。不管是R初学者，还是熟练的R用户都能从书中找到对自己有用的内容。读者既可以把本书作为学习如何应用R的一本优秀教材，也可以作为数据挖掘的工具书。

机械工业出版社（北京市西城区百万庄大街22号　　邮政编码　100037）
责任编辑：盛思源
北京瑞德印刷有限公司印刷
2013年4月第1版第1次印刷
185mm×260mm·13.5印张
标准书号：ISBN 978-7-111-40700-3
定　价：49.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991　88361066　　投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzjsj@hzbook.com

出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

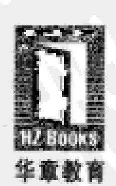
近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短的现状下，美国等发达国家在其计算机科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章公司较早意识到“出版要为教育服务”。自 1998 年开始，我们就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与 Pearson，McGraw-Hill，Elsevier，MIT，John Wiley & Sons，Cengage 等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出 Andrew S. Tanenbaum，Bjarne Stroustrup，Brain W. Kernighan，Dennis Ritchie，Jim Gray，Afred V. Aho，John E. Hopcroft，Jeffrey D. Ullman，Abraham Silberschatz，William Stallings，Donald E. Knuth，John L. Hennessy，Larry L. Peterson 等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及珍藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专程为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近两百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证。随着计算机科学与技术专业学科建设的不断完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

华章网站：www.hzbook.com
电子邮件：hzjsj@hzbook.com
联系电话：（010）88379604
联系地址：北京市西城区百万庄南街 1 号
邮政编码：100037



华章科技图书出版中心

推　荐　序

数据挖掘正在改变着企业和其他大型组织与客户的互动方式，同时也改变着它们管理复杂过程的方式。大量的数据正在很好地用于预测客户行为和结果。在软件方面，R 以其强大的功能和诱人的价格（免费）正在改变着定量分析的“生态系统”。

本书的目的是引领读者迅速地进入这两个世界。本书以实际案例的方式介绍数据挖掘和 R 软件，这样读者就可以在真实情境中进行学习，而不会迷失在统计理论的细节讨论或者计算机科学的基础概念中。本书中用到的工具全部是免费的：MySQL 数据库（用于数据库操作）和 R 软件（用于分析）。因此，本书教给你的是如何动手的知识。通过学习本书，你将体验到数据挖掘和 R 的强大功能。如果你能安装这些工具，并通过应用这些工具来详细地学习书中的案例研究，你将收获颇丰。本书逐步地通过案例研究来介绍 R 的概念，如果你还不熟悉 R 或者 MySQL，你可以按章节顺序来学习这些案例。

本书的原作者 Luís Torgo，根据他在葡萄牙波尔图大学丰富的教学经验、在其他国家讲授数据挖掘课程的经验，以及聚集了世界各地专业人士的 Statistics. com 在线课程中的教学经验，精心地写作了本书。

2012 年 12 月 17 日

Statistics. com 在线课程网站总裁 Peter Bruce



中文版序

目前，数据挖掘和 R 是学术界及工业界中的两个关键技术。丰富的传感器机制使得自动收集数据成为可能后，产生了非常大的数据集，这需要自动化的机制来将这些数据转化为有用的信息，以供决策者使用和参考。R 是一个开发这些自动化机制的很好选择。R 提供的大量算法和方法，以及它的自由和开放源码特性，使得 R 成为数据挖掘的最佳选择之一。本书的目的是向读者介绍数据挖掘和 R 的知识。本书的写作思路是给读者介绍一系列有代表性的研究案例，通过这些案例，读者不仅从中学到主流的数据挖掘方法，同时也可以学习本书所提供的 R 代码，并最终把这些代码应用到他们自己的数据挖掘项目中。

随着中文版的出版，我希望我能说服更多的人认识 R 和数据挖掘的优势。得知我的书得到世界各地读者的关注，对我而言是一项伟大的荣誉。我相信本书中文版的发行将有助于中国的 R 社区。对所有的中国读者，我真诚地希望，在读完本书后，你们发现它不仅有助于你们的工作，同时你们将和我自己一样增加了对数据挖掘和 R 的热情。

Luís Torgo

2012 年 12 月 16 日于葡萄牙，波尔多



译 者 序

Data Mining with R：Learning with Case Studies

本书是2011年查普曼和霍尔公司（Chapman & Hall/CRC）出版的《Data Mining with R：Learning with Case Studies》一书的中文版。英文版从出版后就在亚马逊美国网站上得到了极高的评价，是2011年亚马逊网站上数据挖掘类书籍销量最好的一本。机械工业出版社以极快的速度引进这本书的中文版，使国内读者在原版出版一年左右的时间里读到本书，不得不赞扬他们独到的眼光。本书翻译完稿的时候（2012年10月），其英文版的销量还是排在专业书籍的前列，原作者为本书维护了一个网站，读者可以访问该网站查看这些信息。

本书的作者 Luís Torgo 是一位数据挖掘专家，同时也是一位 R 开发者。本书给出了四个数据挖掘的实际案例，它们分别是藻类频率的预测、证券趋势预测和交易系统仿真、交易欺诈预测，以及微阵列数据分类。这四个案例基本覆盖了常见的数据挖掘技术，从无监督的数据挖掘技术、有监督的数据挖掘技术到半监督的数据挖掘技术。同时这四个案例从数据量、分析目标和数据类型方面引出了各种各样的挑战性问题，本书给出了克服这些挑战的方法和技巧。阅读本书不需要具备 R 和数据挖掘的基础知识。为了便于读者阅读，本书第1章给出了 R 软件的基础知识（安装、R 数据结构、R 编程、R 的输入和输出等）。全书以实际问题、解决方案和对解决方案的讨论为主线来组织内容。读者既可以把本书作为学习如何应用 R 的一本优秀教材，也可以作为数据挖掘的工具书。读者可以根据自己的需要参考书中的某些具体方法，找到自己实际问题的解决方案。

R 本身是一款十分优秀的统计分析和数据挖掘软件，有关 R 的书籍和文档也是相当多的。但是系统地讲解用 R 进行数据挖掘的书籍目前还没有。本书以四个案例研究的形式组织内容，脉络清晰，并且各章自成体系。读者可以从头逐章学习，也可以根据自己的需要进行学习。不管是 R 初学者，还是熟练的 R 用户都能从书中找到对自己有用的内容。

本人在2011年年初学习作者 Luís Torgo 在 Statistics. com 上的在线课程，深感本书的内容极具实用价值，萌生了把本书翻译为中文的念头。2011年年末，恰逢机械工业出版社华章公司引进了本书的版权，在王春华编辑的支持下，我承担了本书的翻译工作。由于英文的习惯和汉语有较大的不同，对于一些特别长的句式，译者按照原文的意思进行了分解处理。关于书中的术语，译者尽量采用中文已有的对应术语，如果中文没有对应术语，译者尽力采用贴切的名称来反映原文中的术语。

本书的翻译工作由李洪成、陈道轮和吴立明共同完成。另外，许金玮、朱振兴、陈冰、汤静文、瞿秋霞、张潇予等也对本书的部分翻译提供了帮助。在本书的翻译过程中，原作者 Torgo 博士多次就译者提出的问题进行耐心而细致的解答。这里对他的帮助表示由衷的谢意。另外，感谢美国统计教育学院 Peter Bruce 为本书中文版写的推荐序。由于水平所限，书中可能会有翻译不当之处，希望读者多加指正。

李洪成

本书的主要目的是向读者介绍如何用R进行数据挖掘。R是一个可以自由下载[⊖]的语言，它提供统计计算和绘图环境，其功能和大量的添加包使它成为一款优秀的、多个已有（昂贵）数据挖掘工具的替代软件。

数据挖掘的一个关键问题是数据量。典型的数据挖掘问题包括一个大的数据库，需要从中提取有用的信息。在本书中，我们用MySQL作为核心数据库管理系统。对多个计算机平台，MySQL也是免费的[⊜]。这意味着，我们可以不用付任何费用就可以进行“重要的”数据挖掘任务。同时，我们希望说明解决方案质量上并没有任何损失。昂贵的工具并不意味着一定更好！只要你愿意花时间来学习如何应用它们，R和MySQL就是一对很难超越的工具。我们认为这是值得的，希望在读完本书之后，你也相信这点。

本书的目的不是介绍数据挖掘的各个方面。许多已有的书籍覆盖了数据挖掘领域。我们用几个案例来向读者介绍R的数据挖掘能力。显然，这几个案例不能代表我们在现实世界中碰到的所有数据挖掘问题。同时，我们给出的解决方案也不是最完全的方案。我们的目的是通过这些实际案例向读者介绍如何用R进行数据挖掘。因此，我们案例分析的目的是展示用R进行信息提取的例子，而不是提供数据挖掘案例的完整分析报告。它们可以作为任何数据挖掘项目的可能思路，或者作为开发数据挖掘项目解决方案的基础。尽管如此，我们尽力尝试覆盖多方面的问题，展示数据大小、不同数据类型、分析目标和进行分析所必需的工具所带来的挑战。然而，这里的实践方式也是有代价的。实际上，作为具体案例研究的一种形式，为了让读者在自己的计算机上执行我们所描述的步骤，我们也做了某些妥协。也就是说，我们不能处理太大的问题，这些问题要求的计算机资源不是每个人都具备的。尽管这样，我们认为本书涵盖的问题也不算小，并对不同的数据类型和维度给出了解决方案。

这里并不要求读者具有R的先验知识。没有学过R和数据挖掘的读者应该可以学习书中的案例。书中的各个案例相互独立，读者可以从书中任何一个案例开始。在第一个简单案例中，给出了一些基本的R知识。这意味着，如果你没有学过R，至少应该从第一个案例开始学习。而且，第1章给出了R和MySQL的简介，它可以帮助你理解后面的章节。我们也没有假设你熟悉数据挖掘和统计技术。在每个案例的必要地方，都对不同的数据挖掘技术进行了介绍。本书的目的不是向读者介绍这些技术的理论细节和全面知识，我们对这些工具的描述包括了它们的基本性质、缺点和分析目标。如果需要进一步了解技术细节，可以参考其他书籍。在某些节的末尾，我们提供了“参考资料”，如果需要，可以参考它们。总之，本书的读者应该是数据分析工具的用户，而不是研究人员或者开发人员。同时，我们希望后者把本书作为进入R和数据挖掘“世界”的一种方式，从而发现本书的用途。

⊖ 下载网址：http://www.r-project.org。

⊜ 下载网址：http://www.mysql.com。

本书有一个免费的 R 代码集，可以从本书网站下载[⊖]。其中含有案例研究中的所有代码，这可以帮助你的实践学习。我们强烈建议读者在阅读本书时安装 R 并实验书中的代码。而且，我们创建了一个名为 DMwR 的 R 添加包，它包含本书用到的多个函数和以 R 格式保存的案例数据集。你应该按照本书的指示，安装并加载该添加包（第 1 章给出了细节）。

⊖ 下载网址：http：//www. liaad. up. pt/~ltorgo/DataMiningWithR/。

首先要感谢我的家人给我的所有支持。没有他们的帮助和支持，我是无法完成本书的。他们的支持、关怀和爱给我足够的安慰，使我可以克服写作本书过程中的困难。同样，我的朋友也给了我同样的安慰，他们在需要时总是和我共同畅饮和交流，带给了我轻松愉悦的写作心情。谢谢我的家人和朋友！谢谢你们！现在，我希望有更多的时间和你们一起分享。

这里我也要感谢我的所有同事和 LIAAD/INESC Porto LA 研究所对我的支持。同时，我也要感谢波尔多大学对我研究的支持。写作本书的部分资助来自于葡萄牙自然科学基金（资助号：SFRH/BSAB/739/2007）。

最后，感谢所有帮助阅读本书草稿的同事和学生。

Luís Torgo

葡萄牙，波尔多



简　介

R是一种用于统计计算的编程语言和环境，它与AT&T贝尔实验室的Rick Becker、John Chambers和Allan Wilks开发的S语言很相似。随着操作系统的不同，R语言有UNIX版本、Windows版本和Mac版本。R可以在不同体系结构的计算机系统上运行，例如Intel、PowerPC、Alpha系统以及Sparc系统。R最早由新西兰奥克兰大学的Ihaka和Gentleman于1996年开发。现在R的开发由一个几十人组成的核心团队来负责，核心团队的成员来自于世界各地的不同机构和单位。由于R的开源性，R软件的开发采用社区合作的方式。R的所有源代码都可以免费获取以便进行检验或者利用。这样你就可以检查和测试你应用的R软件的可靠性。人们对开源软件模式有诸多指责，其中最多的是认为开源软件缺少支持是其主要缺陷之一。但是对于R软件，该缺陷却不存在！有许多很好的文档、书籍和网站提供了免费的R资料。另外，R帮助邮件列表是获取免费R帮助信息和建议的极好来源，它甚至比付费的帮助更好！R有一个可搜索的邮件列表文档。在邮件列表中提问前，可以先查找这个可搜索的邮件列表文档。（也应该这样做！）可以在R网站的“Mailing Lists”（邮件列表）部分得到有关R邮件列表的更多信息。

*数据挖掘*是应用统计学、机器学习和模式识别等学科的知识，从数据中发现有用的、有效的、未知的并且可以理解的信息的一项技术。数据挖掘的一项重要特征是数据的维度。随着计算机技术和信息系统的广泛应用，需要探索的数据呈指数增长。这给传统的数据分析学科带来了挑战：必须考虑计算的效率、内存资源的限制、数据库接口等。这些使得数据挖掘成为一门高度交叉的学科，它不仅有传统数据分析的任务，也有数据库的工作，高维数据可视化等。

由于R的所有计算是在计算机的内存中进行的，所以R在处理大数据集上有限制。但是这并不意味着我们不能处理这些问题，利用R高度灵活的数据库接口，我们可以对大型问题进行数据挖掘。基于开源软件的信念，我们可以应用优秀的MySQL数据库管理系统[⊖]。MySQL可以应用在大多数的计算机平台和操作系统上。而且，R的添加包RMySQL（James and DebRoy, 2009）可以使我们便捷地访问MySQL数据库。

总之，我们希望在读完本书之后，你能够相信不用花钱就可以进行大型问题的数据挖掘。这一切都归功于开发出R和MySQL这样优秀软件的人们的慷慨贡献。

1.1　如何阅读本书

本书基于做中学的宗旨，组织了一系列的案例研究。这些案例的“解决方案”通过R来获取。本书描述了所有得到“解决方案”的必要步骤。通过本书提供的网站[⊜]，可以获取本书有关的R添加包（DMwR），所有的R代码和案例研究数据都在相关的文档中。这可以让你方便地自己进行实验。理论上，你应该在计算机上阅读这些文档并尝试这些文档中演示给你的每一个步

⊖ 下载地址：http：//www.mysql.com。

⊜ http：//www.liaad.up.pt/~ltorgo/DataMiningWithR/。

骤。在本书中，R 代码是用以下代码体来表示的：

```
> R.version
               _
platform       i486-pc-linux-gnu
arch           i486
os             linux-gnu
system         i486, linux-gnu
status
major          2
minor          10.1
year           2009
month          12
day            14
svn rev        50720
language       R
version.string R version 2.10.1 (2009-12-14)
```

[2] 在 R 的命令行提示符“>”之后输入 R 命令。当看到这个提示符时，你可以理解为 R 在等待输入命令。在命令提示符后输入命令，然后按下 ENTER（回车）键让 R 来执行它们。这可能会产生某种形式的输出（即 R 命令的结果），然后出现一个新的提示符。在提示符处，你可以使用箭头键来浏览和编辑以前输入过的命令。若以前输入了类似的命令，通过编辑以前的命令可以方便地得到你需要的命令，这就避免了重复输入。可以复制、粘贴本书网站提供的代码到 R 编译器或者 R 控制台，这样就可以避免自己输入书中的那些代码，这绝对会使你的学习更加便捷，并加深你对书中知识的理解。

1.2 R 简介

本节提供了 R 语言中关键术语的简要介绍。读者不需要熟悉计算机编程知识就可以容易地掌握本节中介绍的例子。不过，如果你觉得没有动力来学习这部分 R 简介，可以先跳过本章的 R 简介部分，直接进行案例研究。当你从具体应用中得到更多的学习 R 语言的动机后，再返回本章的 R 简介部分。

R 是一门用于统计计算和绘图的函数语言。它可以看做是 S 语言的一种方言（由 AT&T 公司开发）。S 语言的开发者 John Chambers 因为 S 语言而于 1998 年被授予计算机学会（Association of Computing Machinery，ACM）软件奖，获奖时提到 S 语言“永远改变了人们分析、可视化和操作数据的方式”。

仅仅在命令行交互方式下 R 就已经非常有用了。也可以有更高级的应用 R 软件系统的方式，比如用户可以开发自己的函数来执行系统性的重复任务，甚至可以利用 R 的开源优点来增加或者修改已有的 R 添加包中的功能。

1.2.1 R 起步

在计算机中安装 R 最简单的方法是从 R 网站[⊖]获取一个二进制发行版。从 R 网站的链接中可以找到综合 R 文档网站（Comprehensive R Archive Network，CRAN），在该网站的诸多资源中，可以找到适合你操作系统/架构的二进制发行版。如果喜欢直接从 R 源代码来安装 R 软件，可以从 [3] R 的 CRAN 网站得到从源代码安装 R 的指导文档。

下载适合你操作系统的二进制发行版后，只需要按照其中的说明，一步一步进行即可。对于

⊖ http://www.r-project.org.

R的Windows版本，只需运行下载文件（R-2.10.1-win32.exe）[一]，在之后的菜单中选择所需要的选项。在某些操作系统中，由于缺乏安装软件的权限，可能需要联系系统管理员来完成安装任务。

在Windows操作系统中，只需要双击桌面上相应的图标就可以运行R；而在UNIX操作系统中，则需要在操作系统提示符后输入字母R。它们最终将启动带有命令行提示符“>”的R控制台。

如果想退出R，可以在R提示符下输入指令q()。R将询问是否要保存当前工作空间的内容。如果想在以后恢复当前R退出时的分析内容，应该回答“是”。

虽然R安装后有了强大的功能，但是你可能需要安装一些CRAN上其他的R添加包。在Windows版本中，很容易通过“程序包”菜单来安装R添加包。将计算机连接到互联网后，应该在菜单中选择“安装程序包...”这个选项。选择该选项后，R首先弹出一个CRAN的镜像窗口，让你选择合适的CRAN镜像网站。选择了CRAN镜像网站之后，它会给出另外一个窗口，列出在CRAN网站上可用的R添加包列表。在列表中选择你需要的添加包后，R将下载该添加包并自动安装在R软件系统中。在UNIX版本中，添加包的安装可能会随着R安装过程中图形功能的不同而略有不同。不过，即使不是使用菜单来安装，通过命令行安装添加包也是很简单的[二]。假设需要要下载一个能够提供连接到MySQL数据库功能的添加包，包的名称是RMySQL[三]。只需要在R提示符处输入以下命令：

```
> install.packages('RMySQL')
```

install.packages()函数有许多参数，其中有CRAN镜像库参数repos，通过该参数可以设定离你最近的CRAN镜像[四]。尽管如此，当第一次在R会话中运行该命令时，R会提示你选择需要使用的CRAN镜像库。

你应该做的事情之一是安装本书提供的R添加包，该添加包提供了本书用到的数据集和贯穿全书的多个函数。和安装其他R添加包一样，安装本书提供的添加包的代码如下：

```
> install.packages('DMwR')
```

4

如果你想知道目前在计算机上已经安装的R添加包，可执行以下命令：

```
> installed.packages()
```

这将产生一个很长的输出，输出的每一行包含一个包名、版本信息、所依赖的包等。另外一种获取已经安装的R添加包的方式是用以下命令：

```
> library()
```

尽管它的输出信息不完整，但是却用户友好的。另外一个非常有用的命令是检查CRAN上是否有已安装的R添加包的更新版本：

```
> old.packages()
```

此外，可以使用下面的命令来更新所有已安装的R软件包：

```
> update.packages()
```

R软件有一个集成的帮助系统，可以用它来了解更多的R函数和R软件系统。此外，可以

[一] 实际的文件名随着R版本的不同而不同，这里的文件名是R 2.10.1的文件名。

[二] 注意，这里的R命令在Windows操作系统下也是可以运行的，尽管应用菜单更实用些。

[三] 可以从R的CRAN网站的常见问题部分（R FAQ）了解每一个R添加包的功能。

[四] 所有的CRAN镜像库列表可以从网站http://cran.r-project.org/mirrors.html得到。

在R网站找到更多的R文档。R软件带有一组HTML文件，可以使用Web浏览器读取它们。在Windows版本的R上，可以通过帮助菜单来访问这些网页。另外，也可以在R提示符下执行命令help. start() 来启动浏览器以显示这些HTML帮助页面。另一种得到帮助的形式是使用help()函数。例如，如果需要plot() 函数的帮助信息，可以输入命令“help (plot)” (或者? plot)。如果连接到互联网，那么一个相当强大的功能就是使用RSiteSearch() 函数，用它来搜索邮件列表文档、R手册和R帮助页面中的关键词或短语，例如：

```
> RSiteSearch('neural networks')
```

最后，在Web上还有几个地方可以找到R的各种帮助内容，如网站 http://www.rseek.org/。

1.2.2 R对象

R语言有两个主要概念：对象和函数。R对象可以看做是具有关联名称的存储空间。R中的一切都存储在一个对象中。所有的变量、数据、函数等都是以命名对象的形式存储在计算机的内存中。

函数是一种用来进行某个操作的特殊类型的R对象。它们通常接受一些输入参数，通过执行一系列的操作产生结果（它们通常由其他函数来调用)。R已经有大量的函数可供使用，但稍后还将看到，用户还可以创建新的函数。

5

可以使用赋值运算符把内容存储到对象中。赋值运算符是用一个尖括号和一个减号来表示的（<-）[⊖]：

```
> x <- 945
```

上一条指令的作用是把数值945存储在名为x的对象中。只需要在R提示符后输入对象的名称，就可以看到它的内容[⊜]。

```
> x

[1] 945
```

而在数字945前面的“[1]”可以读作“此行是从对象的第一个元素开始显示的值”。后面我们会看到，这样的显示对于包含多个值的对象（例如向量等）特别有用。

下面给出其他赋值语句的例子。这些例子清楚地说明这是一个“破坏性”的操作。因为在任何时间t，一个对象只能有一个给定的内容。这意味着，将某些新的内容分配给现有对象时，该对象就失去了其先前的内容。

```
> y <- 39
> y

[1] 39

> y <- 43
> y

[1] 43
```

也可以把数值表达式赋给一个对象。在这种情况下，该对象将存储表达式的结果：

```
> z <- 5
> w <- z^2
> w
```

⊖ 这里也可以用“=”作为赋值运算符，但是不建议采用，因为“=”可能与相等测试混淆。

⊜ 如果名称输入不正确，则会得到错误信息，这是应用R时经常发生的一种错误!

```
[1] 25

> i <- (z * 2 + 45)/2
> i
[1] 27.5
```

我们可以这样来看赋值运算：无论运算符右侧是什么，都先计算右侧，然后把计算结果赋给（存储）赋值号左侧的对象。

如果只是想知道某些算术运算的结果，那么不需要把表达式的结果赋给对象。实际上，可以把 R 提示符作为一种计算器：

```
> (34 + 90)/12.5

[1] 9.92
```

创建的每个对象都将存储在计算机内存中，直到删除它。可能通过输入 ls( ) 或 objects( ) 命令列举出当前内存中的对象。如果不再需要一个对象了，可以通过删除它来释放一些内存空间：

```
> ls()

[1] "i" "w" "x" "y" "z"

> rm(y)
> rm(z, w, i)
```

对象名称可以包括任何大写字母、小写字母、数字 0 ~ 9（不能用于名称的开头）以及和字母作用相似的符号“.”。注意，在 R 中的名称是区分大小写的，这意味着 Color 和 color 是两个不同的对象。实际上，这常常是导致初学者遇到“找不到对象”错误的一个常见原因。如果遇到这种类型的错误，应该首先检查出现错误的对象名称的正确性。

1.2.3 向量

向量是 R 中最基本的数据对象。甚至当把单一数字赋给一个对象（例如 x <- 45.3）时，也就创建了一个包含单个元素的向量。所有对象都有模式（mode）和长度属性。模式决定了存储在对象中的数据类型。向量用来存储一组基本类型相同的数据。R 的主要基本数据类型是字符型[⊖]、逻辑型、数值型、复数型。因此，向量可以是字符型、逻辑值型（T、F，或者 FALSE、TRUE）[⊜]、数值型和复数型。一个对象的长度是它含有元素的数量，可以用 length( ) 函数来获取。

6 ~ 7

在大多数情况下，使用长度大于 1 的向量。可以在 R 中使用 c( ) 函数和相应的参数来创建一个向量：

```
> v <- c(4, 7, 23.5, 76.2, 80)
> v

[1]  4.0  7.0 23.5 76.2 80.0

> length(v)

[1] 5

> mode(v)

[1] "numeric"
```

⊖ R 的字符型数据事实上是一组字符，而不是你可能认为的单个字符，这在其他编程语言中常常称为字符串类型。

⊜ 注意到 R 中的名称是大小写敏感的。因此，True 不是一个有效的逻辑值。

一个向量的所有元素都必须属于相同的模式。如果不是，R 将强制执行类型转换。下面就是这样的一个例子：

```
> v <- c(4, 7, 23.5, 76.2, 80, "rrt")
> v

[1] "4"    "7"    "23.5" "76.2" "80"   "rrt"
```

向量的所有元素已转换为字符模式。字符值是由单引号或双引号包含的字符串。

所有向量可以包含一个特殊值，即 NA，该值代表缺失值：

```
> u <- c(4, 6, NA, 2)
> u

[1]  4  6 NA  2

> k <- c(T, F, NA, TRUE)
> k

[1]  TRUE FALSE    NA  TRUE
```

通过方括号之间的索引，可以访问一个向量的某个特定元素：

```
> v[2]

[1] "7"
```

上例给出了向量 *v* 的第二个元素。在后面的 1.2.7 节中，我们将学习使用索引向量来获得更强大的索引方法。

8

通过使用相同的索引策略，可以改变一个特定向量元素的值。

```
> v[1] <- "hello"
> v

[1] "hello" "7"     "23.5"  "76.2"  "80"    "rrt"
```

R 允许创建空向量：

```
> x <- vector()
```

如果使用不存在的索引来添加向量元素，就可以改变向量的长度。例如，创建空向量 *x* 后，可以输入

```
> x[3] <- 45
> x

[1] NA NA 45
```

注意，向量的前两个元素有未知的值 NA。这种灵活性是有代价的。与其他编程语言不同，如果在 R 中使用一个不存在的向量位置，不会得到错误：

```
> length(x)

[1] 3

> x[10]

[1] NA

> x[5] <- 4
> x

[1] NA NA 45 NA  4
```

为了缩小向量的大小，可以利用之前提到过的赋值运算的破坏性性质，例如：

```
> v <- c(45, 243, 78, 343, 445, 44, 56, 77)
> v

[1]  45 243  78 343 445  44  56  77

> v <- c(v[5], v[7])
> v

[1] 445  56
```

尽管将在1.2.7节学习使用更强大的索引方法，但这里仍然可以用一个简单的方法删除向量的特定元素。

9

1.2.4 向量化

R语言最强大的方面之一就是函数的向量化。这些函数可以直接对向量的每个元素进行操作。例如：

```
> v <- c(4, 7, 23.5, 76.2, 80)
> x <- sqrt(v)
> x

[1] 2.000000 2.645751 4.847680 8.729261 8.944272
```

sqrt() 函数计算其参数的算术平方根。这种情况下，使用数值向量作为它的参数。向量化函数输出一个与输入参数相同长度的结果向量，结果向量的每个元素是函数应用到原来输入向量的相应元素得到的结果。

也可以利用R的这个特性进行向量的算术运算：

```
> v1 <- c(4, 6, 87)
> v2 <- c(34, 32.4, 12)
> v1 + v2

[1] 38.0 38.4 99.0
```

如果两个向量的长度不同，向量应该如何运算呢？R将使用*循环规则*，该规则重复较短的向量元素，直到得到的向量长度与较长向量的长度相同。例如：

```
> v1 <- c(4, 6, 8, 24)
> v2 <- c(10, 2)
> v1 + v2

[1] 14  8 18 26
```

它是把向量c(10,2)扩充为c(10,2,10,2)。如果较长向量的长度不是较短向量的整数倍，则R给出警告：

```
> v1 <- c(4, 6, 8, 24)
> v2 <- c(10, 2, 4)
> v1 + v2

[1] 14  8 12 34
Warning message:
In v1 + v2 :
  longer object length is not a multiple of shorter object length
```

循环规则已经应用，运算也完成了。(这里只是给出一个警告，而不是错误!)

如前所述，单个数字在R中表示为长度为1的向量。这种表示在下面的运算中非常方便：

10

```
> v1 <- c(4, 6, 8, 24)
> 2 * v1

[1]  8 12 16 48
```

注意，数字 2（实际上是向量 c(2)!）被循环，导致 v1 的所有元素乘以 2。正如我们将看到的，这种循环规则也适用于其他的对象，如数组和矩阵。

1.2.5 因子

因子提供了一个简单而又紧凑的形式来处理分类（名义）数据。因子用水平来表示所有可能的取值。如果数据集有取值个数固定的名义变量，因子就特别有用。下面的章节将要学习的多个图形函数和汇总函数就应用了因子的这种优点。对用户来说，这种使用和显示因子数据的方式显然是易于理解的，而 R 软件内部以数值编码方式来存储因子值，这将大大提高内存的利用效率。

下面举例说明如何在 R 中创建因子。假设有一个 10 个人的性别向量：

```
> g <- c("f", "m", "m", "m", "f", "m", "f", "m", "f", "f")
> g

 [1] "f" "m" "m" "m" "f" "m" "f" "m" "f" "f"
```

你可以把这个向量转换为一个因子：

```
> g <- factor(g)
> g

 [1] f m m m f m f m f f
Levels: f m
```

注意，得到的不再是一个字符向量。上面提到，实际上这些因子在 R 内部表示为数值向量[⊖]。在这个例子中，因子有两个水平，'f' 和 'm'，在 R 内部分别表示为 1 和 2。然而，你不需要关心这个内部表示，因为你可以使用“原始的”字符值，R 在显示因子时也使用这种字符方式。因此，出于效率的考虑，R 因子的编码转换是用户透明的。

假设有另外 5 个人，需要把他们的性别信息存储在另一个因子对象中。假设他们都是男性。如果仍然需要这个因子对象与对象 g 有两个相同的因子水平，则必须使用以下命令：

11

```
> other.g <- factor(c("m", "m", "m", "m", "m"), levels = c("f",
+     "m"))
> other.g

[1] m m m m m
Levels: f m
```

如果没有在输入参数中设定 levels 参数，因子 other.g 将只有一个水平（'m'）。

在 R 这样的函数型编程语言中，最常见的应用函数的方式之一就是像上例中的函数复合方式。它是把函数（factor()）应用到另一个函数（c()）的结果。显然，我们可以首先把函数 c() 的结果赋值给一个对象，然后再用该对象调用 factor() 函数。然而，这会更麻烦并且创建多余对象也会浪费内存，因此人们经常使用这种函数复合。函数复合的缺点是，它给不熟悉这种函数复合的用户带来阅读困难。

利用因子类型数据，可以做的事情之一是计算每个可能值的发生次数。例如：

⊖ 可以通过输入 mode(g) 来确认这一点。

```
> table(g)

g
f m
5 5

> table(other.g)

other.g
f m
0 5
```

table( ) 函数也可以用于获取多个因子的交叉表。假设向量 *a* 存储 10 个人所属的年龄，那么可以得到这两个向量的交叉表：

```
> a <- factor(c('adult','adult','juvenile','juvenile','adult','adult',
+               'adult','juvenile','adult','juvenile'))
> table(a,g)

          g
a          f m
  adult    4 2
  juvenile 1 3
```

12

你可能已经注意到有的行是以一个"+"号开始，当一行代码太长，你决定在命令完成之前另起一个新行（按"回车"键）时，就会使用"+"号，这时候由于上一行没有完，所以 R 在新起的一行以"+"开始，它是新行继续提示符。要记住的是，新行继续提示符不是输入的！它们是由 R（如正常提示符">"）自动输出的。

有时候，我们希望计算列联表的边际和相对频率。下面给出了上面数据集的性别和年龄因子的总计：

```
> t <- table(a, g)
> margin.table(t, 1)

a
   adult juvenile
       6        4

> margin.table(t, 2)

g
f m
5 5
```

函数的输入参数"1"和"2"分别代表列联表的第一个和第二个维度，即表 t 的行和列。每个维度边际和总计的相对频率如下：

```
> prop.table(t, 1)

          g
a                  f         m
  adult    0.6666667 0.3333333
  juvenile 0.2500000 0.7500000

> prop.table(t, 2)

          g
a            f   m
  adult    0.8 0.4
```

```
  juvenile 0.2 0.6
> prop.table(t)
          g
a           f   m
  adult     0.4 0.2
  juvenile 0.1 0.3
```

13 注意，如果需要的是百分比，可以在调用函数时乘以 100。

1.2.6 生成序列

R 提供了多种生成不同类型序列的方法。比如，创建一个包含 1 ~ 1000 所有整数的向量，可以简单地输入

```
> x <- 1:1000
```

就创建了一个名为 *x* 的向量，向量 *x* 包含了 1000 个元素，即 1 ~ 1000 的所有整数。

注意运算符“:”的优先级，我们通过下面的例子来说明这个问题：

```
> 10:15 - 1
[1]  9 10 11 12 13 14
> 10:(15 - 1)
[1] 10 11 12 13 14
```

这里需要理解第一个命令的结果（记住循环规则），“:”的优先级高于减法“-”。

同样，可以生成如下的递减序列：

```
> 5:0
[1] 5 4 3 2 1 0
```

可以利用函数 seq() 生成实数序列，比如：

```
> seq(-4, 1, 0.5)
 [1] -4.0 -3.5 -3.0 -2.5 -2.0 -1.5 -1.0 -0.5  0.0  0.5  1.0
```

生成了从 -4 ~ 1、以 0.5 为增量的一个实数序列。函数 seq() 还有其他功能[⊖]。下面举例说明 seq() 函数的其他功能：

14

```
> seq(from = 1, to = 5, length = 4)
[1] 1.000000 2.333333 3.666667 5.000000
> seq(from = 1, to = 5, length = 2)
[1] 1 5
> seq(length = 10, from = -2, by = 0.2)
[1] -2.0 -1.8 -1.6 -1.4 -1.2 -1.0 -0.8 -0.6 -0.4 -0.2
```

你可能已经注意到，在上面的例子中，在函数调用中可以有不同的方式给出函数参数——先给出参数名，再给出该参数需要使用的特定参数值。因此，当我们使用带有多个参

⊖ 你可以通过在 R 命令窗口输入“? seq”得到函数 seq() 的帮助信息。通过阅读函数 seq() 的帮助文档，可以更好地理解和掌握函数 seq() 的参数和其他函数形式。

数且大部分参数都采用默认值的函数时，这将非常方便。一旦这些参数默认值满足我们的需要，我们就可以避免人为地指定设置这些参数。但是，如果这些默认值不适用于我们的问题，那么我们就需要提供其他可选值。如果没有像上面例子那样通过参数名设定参数，则需要根据参数位置来设定参数值。如果想要改变的默认值参数是函数的最后参数之一，那么根据参数位置设定将要求对该参数之前所有的参数值进行设定，不管是否使用的是参数的默认值[⊖]。因为在设定参数值时给出了参数的名称，所以按名称设定参数可以在函数调用中改变参数的顺序。

另一个产生具有某种模式序列的有用函数是 rep( ) 函数。比如：

```
> rep(5, 10)

 [1] 5 5 5 5 5 5 5 5 5 5

> rep("hi", 3)

[1] "hi" "hi" "hi"

> rep(1:2, 3)

[1] 1 2 1 2 1 2

> rep(1:2, each = 3)

[1] 1 1 1 2 2 2
```

gl( ) 函数可用于生成带有因子的序列。这个函数的语法是 gl(*k*, *n*)，其中 *k* 是因子水平的个数，*n* 是每个水平的重复数。这里举两个例子：

```
> gl(3, 5)

 [1] 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3
Levels: 1 2 3

> gl(2, 5, labels = c("female", "male"))

 [1] female female female female female male   male   male   male   male
Levels: female male
```

最后，R 有多个可以根据不同概率密度函数来生成随机序列的函数。这些函数的通用结构是 r*func*(n, par1, par2, …)，其中 *func* 是概率分布的名称，n 是要生成的随机数的个数，par1, par2, …是概率密度函数所需要的一些参数值。例如，可以产生 10 个服从均值为 0、标准差为 1 的正态分布的随机数值：

```
> rnorm(10)

 [1] -0.74350857  1.14875838  0.26971256  1.06230562 -0.46296225
 [6] -0.89086612 -0.12533888 -0.08887182  1.27165411  0.86652581
```

为了产生 4 个服从均值为 0、标准差为 3 的正态分布的随机数值，可以使用

```
> rnorm(4, mean = 10, sd = 3)

[1]  5.319385 15.133113  8.449766 10.817147
```

为了获得 5 个服从自由度为 10 的 *t* 分布的随机数值，可以输入

⊖ 实际上，我们可以简单地使用逗号直到达到我们所期望的位置，如 seq(1,4,,40)。

```
> rt(5, df = 10)

[1] -1.2697062  0.5467355  0.7979222  0.4949397  0.2497204
```

R 有更多的概率函数，以及其他获取概率密度、累计概率和这些分布的分位数函数。

1.2.7 数据子集

前面例子中提到，可以在方括号内放入元素的位置来获得向量中的某个元素。R 也允许在方括号中使用向量。R 有多种类型的索引向量。逻辑索引向量可以提取相应于真值的元素。让我们来看一个具体的例子：

15 ~ 16

```
> x <- c(0, -3, 4, -1, 45, 90, -5)
> x > 0

[1] FALSE FALSE  TRUE FALSE  TRUE  TRUE FALSE
```

上面显示的第二个命令是逻辑条件。由于 *x* 是向量，所以将向量中所有的元素与 0 进行比较，（这里应用了循环规则！）产生一个长度与向量 *x* 相同的逻辑值向量。如果使用该逻辑值向量对 *x* 进行索引，就可以得到相应 TRUE 值位置的向量 *x* 的元素：

```
> x[x > 0]

[1]  4 45 90
```

上面代码可以这样理解：给出逻辑表达式为真的 *x* 的位置。注意，这是应用函数复合的另一个例子，函数复合在本书中应用得相当频繁。利用 R 中的逻辑运算符，可以使用更复杂的逻辑索引向量，例如：

```
> x[x <= -2 | x > 5]

[1] -3 45 90 -5

> x[x > 40 & x < 100]

[1] 45 90
```

你可能已经猜到："|" 运算符表示逻辑或；"&" 运算符表示逻辑与[⊖]。这表明，第一个命令要求小于等于 -2，或者大于 5 的 *x* 的元素。第二个例子表示大于 40 同时小于 100 的 *x* 中的所有元素。

R 还可以使用整数向量来提取向量中的多个元素，索引向量中的数字表示提取的元素在原向量中的位置。例如，

17

```
> x[c(4, 6)]

[1] -1 90

> x[1:3]

[1]  0 -3  4

> y <- c(1, 4)
> x[y]

[1]  0 -1
```

⊖ R 也有替代上面单运算符的双运算符，即 && 和 | |，它们执行相同的运算。这两个替代运算符从左到右来计算表达式，并且只检验向量的第一个元素，而单字符运算符则检验向量的每个元素。

另外，也可以使用一个负值的索引向量来表示哪些元素可以排除。例如：

```
> x[-1]
[1] -3  4 -1 45 90 -5
> x[-c(4, 6)]
[1]  0 -3  4 45 -5
> x[-(1:3)]
[1] -1 45 90 -5
```

注意，在上面的第三个例子中，由于运算符“:”的优先级较高，所以我们使用了括号。

可以通过 R 函数 names( ) 来给向量中的元素命名。对于命名的向量元素，可以通过字符串向量来进行索引。由于命名的元素位置更容易记住，所以有时候命名元素更受欢迎。例如，在5个不同的地方测量了一个化学参数的测量值向量。可以创建如下的命名向量：

```
> pH <- c(4.5, 7, 7.3, 8.2, 6.3)
> names(pH) <- c("area1", "area2", "mud", "dam", "middle")
> pH
 area1  area2    mud    dam middle
   4.5    7.0    7.3    8.2    6.3
```

实际上，如果在创建向量时，已经知道向量中位置的名称，那么下面的方式更简单：

```
> pH <- c(area1 = 4.5, area2 = 7, mud = 7.3, dam = 8.2, middle = 6.3)
```

现在可以使用上面给出的名称对 pH 向量进行索引：

```
> pH["mud"]
mud
7.3
> pH[c("area1", "dam")]
area1   dam
  4.5   8.2
```

最后，当索引为空时，表示所有的元素都被选定。空索引表示是在选择过程中没有限制条件。例如，需要给一个向量赋予零值，那么可以简单地写成“x[ ] <- 0”。注意，这与“x <- 0”不同。后一种情况表示把含有单一元素（零）的向量赋给 *x*，而前面一种情况（假设之前存在向量 *x*）将会使得向量 *x* 的所有元素都变成零。试一下这两种代码！ 18

1.2.8 矩阵和数组

数据元素可以保存在具有多个维度的对象中。在多种情况下这尤其有用。数组存储的是多维数据元素。矩阵是数组的特殊情况，它具有两个维度。在 R 中，数组和矩阵都是带有维度这个特定属性的向量。假设有一个数值向量 *c*(45, 23, 66, 77, 33, 44, 56, 12, 78, 23)，下面把这10个数值组织为一个矩阵：

```
> m <- c(45, 23, 66, 77, 33, 44, 56, 12, 78, 23)
> m
 [1] 45 23 66 77 33 44 56 12 78 23
> dim(m) <- c(2, 5)
> m
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]   45   66   33   56   78
[2,]   23   77   44   12   23
```

注意，数值如何分配到这个 2 行 5 列的矩阵中（这里用函数 dim( ) 来给维度 m 赋值）。其实，可以使用更简单的命令来创建该矩阵：

```
> m <- matrix(c(45, 23, 66, 77, 33, 44, 56, 12, 78, 23), 2,
+     5)
```

你可能已经注意到，向量中的数据通过矩阵中的列进行扩展。首先，将数据填到第一列，然后填到第二列，以此类推。可以通过设定函数 matrix( ) 的参数来按行填充矩阵：

```
> m <- matrix(c(45, 23, 66, 77, 33, 44, 56, 12, 78, 23), 2,
+     5, byrow = T)
> m

     [,1] [,2] [,3] [,4] [,5]
[1,]   45   23   66   77   33
[2,]   44   56   12   78   23
```

如矩阵的显示所示，可以通过类似于向量中的索引方法来访问矩阵的元素，但现在需要两个索引（矩阵的维度）：

```
> m[2, 3]

[1] 12
```

19

可以利用 1.2.7 节所描述的索引方法来提取矩阵中的元素，如下面的例子所示：

```
> m[-2, 1]

[1] 45

> m[1, -c(3, 5)]

[1] 45 23 77
```

此外，如果忽略了任一维度，将会得到矩阵的所有行或列：

```
> m[1, ]

[1] 45 23 66 77 33

> m[, 4]

[1] 77 78
```

注意，在上面两个例子中，作为数据子集，得到的结果可能是一个向量。如果需要得到的结果仍然是一个矩阵，那么即使它是一个 1 行或者 1 列的矩阵，都可以使用下面的命令：

```
> m[1, , drop = F]

     [,1] [,2] [,3] [,4] [,5]
[1,]   45   23   66   77   33

> m[, 4, drop = F]

     [,1]
[1,]   77
[2,]   78
```

函数 cbind( ) 和 rbind( ) 可以分别按列和行用把两个或两个以上的向量或矩阵合并在一

起。我们通过下面的例子来说明这一点：

```
> m1 <- matrix(c(45, 23, 66, 77, 33, 44, 56, 12, 78, 23), 2,
+      5)
> m1

     [,1] [,2] [,3] [,4] [,5]
[1,]   45   66   33   56   78
[2,]   23   77   44   12   23

> cbind(c(4, 76), m1[, 4])

     [,1] [,2]
[1,]    4   56
[2,]   76   12

> m2 <- matrix(rep(10, 20), 4, 5)
> m2

     [,1] [,2] [,3] [,4] [,5]
[1,]   10   10   10   10   10
[2,]   10   10   10   10   10
[3,]   10   10   10   10   10
[4,]   10   10   10   10   10

> m3 <- rbind(m1[1, ], m2[3, ])
> m3

     [,1] [,2] [,3] [,4] [,5]
[1,]   45   66   33   56   78
[2,]   10   10   10   10   10
```

20

我们也可以使用函数 colnames( ) 和 rownames( ) 分别给矩阵的行和列命名，这样有利于记住数据的位置。

```
> results <- matrix(c(10, 30, 40, 50, 43, 56, 21, 30), 2, 4,
+      byrow = T)
> colnames(results) <- c("1qrt", "2qrt", "3qrt", "4qrt")
> rownames(results) <- c("store1", "store2")
> results

       1qrt 2qrt 3qrt 4qrt
store1   10   30   40   50
store2   43   56   21   30

> results["store1", ]

1qrt 2qrt 3qrt 4qrt
  10   30   40   50

> results["store2", c("1qrt", "4qrt")]

1qrt 4qrt
  43   30
```

数组是矩阵的扩展，它把数据的维度扩展到两个以上。这意味着数组中的元素需要两个以上的索引。除此之外，数组与矩阵类似，可以用相同的方法使用。与函数 matrix( ) 类似，可以通过函数 array( ) 方便地创建数组。下面是应用函数 array( ) 的一个例子：

21

```
> a <- array(1:24, dim = c(4, 3, 2))
> a

, , 1

     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12

, , 2

     [,1] [,2] [,3]
[1,]   13   17   21
[2,]   14   18   22
[3,]   15   19   23
[4,]   16   20   24
```

我们可以使用与向量索引同样的方法访问数组中的元素，如下面例子所示。

```
> a[1, 3, 2]

[1] 21

> a[1, , 2]

[1] 13 17 21

> a[4, 3, ]

[1] 12 24

> a[c(2, 3), , -2]

     [,1] [,2] [,3]
[1,]    2    6   10
[2,]    3    7   11
```

尽管有时候理解起来有些困难，但循环规则和算术运算规则同样适用于矩阵和数组。下面是几个例子：

```
> m <- matrix(c(45, 23, 66, 77, 33, 44, 56, 12, 78, 23), 2,
+       5)
> m.

     [,1] [,2] [,3] [,4] [,5]
[1,]   45   66   33   56   78
[2,]   23   77   44   12   23

> m * 3

     [,1] [,2] [,3] [,4] [,5]
[1,]  135  198   99  168  234
[2,]   69  231  132   36   69

> m1 <- matrix(c(45, 23, 66, 77, 33, 44), 2, 3)
> m1
```

```
     [,1] [,2] [,3]
[1,]   45   66   33
[2,]   23   77   44

> m2 <- matrix(c(12, 65, 32, 7, 4, 78), 2, 3)
> m2

     [,1] [,2] [,3]
[1,]   12   32    4
[2,]   65    7   78

> m1 + m2

     [,1] [,2] [,3]
[1,]   57   98   37
[2,]   88   84  122
```

R 也包含了标准的矩阵代数运算操作和函数，可以阅读 R 随机文档“R 简介”的第五节以获取矩阵运算的更多信息。

1.2.9 列表

R 列表是以其他对象为成分的有序集合。列表的成分和向量的元素不同，它们不一定是同一种数据类型、模式或者相同长度。列表的成分总是编号的并且有一个名称属性。下面用一个简单例子来说明如何构造一个列表：

```
> my.lst <- list(stud.id=34453,
+                stud.name="John",
+                stud.marks=c(14.3,12,15,19))
```

对象 my. lst 由三个成分组成：第一个是名称为 stud. id 的数值；第二个是名称为 stud. name 的字符串；第三个是名称为 stud. marks 的数值向量。

和查看其他 R 对象一样，通过输入列表的名称来查看该列表的内容，例如：

```
> my.lst

$stud.id
[1] 34453

$stud.name
[1] "John"

$stud.marks
[1] 14.3 12.0 15.0 19.0
```

可以通过下面的索引方式得到列表的元素：

```
> my.lst[[1]]

[1] 34453

> my.lst[[3]]

[1] 14.3 12.0 15.0 19.0
```

注意，上例中应用了双方括号。如果应用命令 my. lst[1]，那么将会得到不同的结果：

```
> my.lst[1]

$stud.id
[1] 34453
```

第二个命令获取列表 my. lst 的第一个成分构成的子列表。相反，my. lst[[1]] 提取列表的第一个组件的值（在这种情况下是一个数），结果将不再是一个列表，如下所示：

```
> mode(my.lst[1])

[1] "list"

> mode(my.lst[[1]])

[1] "numeric"
```

对于含有命名成分的列表（如上例所示），可以用另一种方式来提取列表成分的值：

```
> my.lst$stud.id

[1] 34453
```

22 ~ 24

列表成分的名称实际上是列表的一个属性，它可以像向量元素名那样进行操作：

```
> names(my.lst)

[1] "stud.id"    "stud.name"  "stud.marks"

> names(my.lst) <- c("id", "name", "marks")
> my.lst

$id
[1] 34453

$name
[1] "John"

$marks
[1] 14.3 12.0 15.0 19.0
```

也可以通过添加附加元素的方式来扩展列表：

```
> my.lst$parents.names <- c("Ana", "Mike")
> my.lst

$id
[1] 34453

$name
[1] "John"

$marks
[1] 14.3 12.0 15.0 19.0

$parents.names
[1] "Ana"  "Mike"
```

可以使用函数 length( ) 来检查列表成分的个数：

```
> length(my.lst)

[1] 4
```

也可以如下所示来剔除列表的成分：

```
> my.lst <- my.lst[-5]
```

也可以通过函数 c( ) 来合并列表：

```
> other <- list(age = 19, sex = "male")
> lst <- c(my.lst, other)
> lst

$id
[1] 34453

$name
[1] "John"

$marks
[1] 14.3 12.0 15.0 19.0

$parents.names
[1] "Ana"  "Mike"

$age
[1] 19

$sex
[1] "male"
```

最后，可以通过函数 unlist( ) 把列表中的所有元素转换为向量元素，转换后的向量元素的个数和列表中所有数据对象的个数相同。该操作将把列表中不同类型的数据转为统一的类型[⊖]，这意味着大多数情况下把所有列表数据转换为字符型。另外，得到的向量元素都有一个源自列表成分的名称：

```
> unlist(my.lst)

          id            name          marks1        marks2         marks3
     "34453"          "John"          "14.3"          "12"           "15"
      marks4 parents.names1 parents.names2
        "19"           "Ana"          "Mike"
```

1.2.10 数据框

数据框是 R 软件中用于存储数据表的一种数据结构。它的结构与二维矩阵类似。然而，与矩阵不同的是，数据框的每列可以有不同数据类型的数据。在这个意义上，数据框和列表更相似。实际上，对 R 而言，数据框是一类特殊的列表。

可以把数据框的每一行作为一个观测值（或称为个案），它由一组变量（数据框的命名列）来描述。

可以用下列方式来创建数据框：

```
> my.dataset <- data.frame(site=c('A','B','A','A','B'),
+ season=c('Winter','Summer','Summer','Spring','Fall'),
+ pH = c(7.4,6.3,8.6,7.2,8.9))
> my.dataset
  site season  pH
1    A Winter 7.4
2    B Summer 6.3
3    A Summer 8.6
4    A Spring 7.2
5    B   Fall 8.9
```

⊖ 因为向量元素的类型必须相同，参见 1.2.3 节。

可以像矩阵那样访问数据框的元素：

```
> my.dataset[3, 2]

[1] Summer
Levels: Fall Spring Summer Winter
```

注意，“season” 列的所有元素为字符型，因而该列被转换为因子类型。类似地，“site” 列也是因子类型。这些转换是函数 data. frame( ) 的默认行为[⊖]。

1.2.7 节给出的索引方法也适用于数据框。另外，也可以用列名来获取数据框的一列数据：

```
> my.dataset$pH

[1] 7.4 6.3 8.6 7.2 8.9
```

可以利用 R 的数据子集的优势来方便地访问数据框中的数据，如下所示：

```
> my.dataset[my.dataset$pH > 7, ]

  site season  pH
1    A Winter 7.4
3    A Summer 8.6
4    A Spring 7.2
5    B   Fall 8.9

> my.dataset[my.dataset$site == "A", "pH"]

[1] 7.4 8.6 7.2

> my.dataset[my.dataset$season == "Summer", c("site", "pH")]

  site  pH
2    B 6.3
3    A 8.6
```

25
~
27

可以通过应用函数 attach( ) 来简化上面的查询。函数 attach( ) 可以直接访问数据框的列，而无需添加相应的数据框名：

```
> attach(my.dataset)
> my.dataset[site=='B', ]

  site season  pH
2    B Summer 6.3
5    B   Fall 8.9

> season

[1] Winter Summer Summer Spring Fall
Levels: Fall Spring Summer Winter
```

函数 attach( ) 的反向操作是函数 detach( )，它禁止直接访问数据框的列。

```
> detach(my.dataset)
> season

Error: Object "season" not found
```

如果仅对数据框进行简单的查询，那么应用函数 subset( ) 将十分方便：

⊖ 参考函数 data frame( ) 帮助文档中应用函数 I( ) 的例子，或者应用参数 stringsAsFactors 来避免这种强制转换。

```
> subset(my.dataset, pH > 8)

  site season  pH
3    A Summer 8.6
5    B   Fall 8.9

> subset(my.dataset, season == "Summer", season:pH)

  season  pH
2 Summer 6.3
3 Summer 8.6
```

和上面的例子不同，不能用取子集的方式来改变数据框中数据的值。例如，如果需要把季节为“summer”的行的 pH 值加 1，只能通过下列方式进行：

```
> my.dataset[my.dataset$season == 'Summer','pH'] <-
+       my.dataset[my.dataset$season == 'Summer','pH'] + 1
```

与列表类似，可以在数据框中加入新列：

28

```
> my.dataset$NO3 <- c(234.5, 256.6, 654.1, 356.7, 776.4)
> my.dataset

  site season  pH   NO3
1    A Winter 7.4 234.5
2    B Summer 7.3 256.6
3    A Summer 9.6 654.1
4    A Spring 7.2 356.7
5    B   Fall 8.9 776.4
```

对于加入新列的唯一限制就是新列必须和已有的数据框有相同的行数；否则，R 会报错。可以用下列两个函数来获得数据框的行数或列数：

```
> nrow(my.dataset)

[1] 5

> ncol(my.dataset)

[1] 4
```

在数据挖掘的任务中，通常很少需要用上面提到的 data.frame( ) 函数来手动输入数据。一般是把来源于文件或者数据库的数据集读入数据框。在后面的数据挖掘案例研究章节中，你将学习如何把数据导入到数据框中。无论如何，建议你浏览 R 提供的“R 数据导入和导出”手册，了解 R 所具有的功能。

R 有一个类似于电子表格的接口，可以用于小型数据的输入。可以用下面的方式来编辑一个已经存在的数据框：

```
> my.dataset <- edit(my.dataset)
```

或者可以创建一个新的数据框，

```
> new.data <- edit(data.frame())
```

可以用一个名称向量来改变数据框的列名：

```
> names(my.dataset)

[1] "site"   "season" "pH"     "NO3"
```

```
> names(my.dataset) <- c("area", "season", "pH", "NO3")
> my.dataset

  area season  pH   NO3
1    A Winter 7.4 234.5
2    B Summer 7.3 256.6
3    A Summer 9.6 654.1
4    A Spring 7.2 356.7
5    B   Fall 8.9 776.4
```

因为数据框的名称属性是向量，所以如果仅仅需要改变数据框某个特定列的名称，就可以输入：

```
> names(my.dataset)[4] <- "PO4"
> my.dataset

  area season  pH   PO4
1    A Winter 7.4 234.5
2    B Summer 7.3 256.6
3    A Summer 9.6 654.1
4    A Spring 7.2 356.7
5    B   Fall 8.9 776.4
```

最后，R 有一些“内置”的数据集，可以用它们来学习和探索 R 的功能。大多数的 R 添加包也有自带的数据集。为了获取已有数据集的信息，可以执行下列命令：

```
> data()
```

可以通过下列方式来应用已有的数据集：

```
> data(USArrests)
```

上面的命令“创建”了一个名为 USArrests 的数据框，它含有内置在 R 中相应问题的数据集。

1.2.11 构建新函数

R 允许用户构造新函数。当需要自动化某些不断重复的任务时，应用 R 的这一特性将十分有用。每次需要执行任务时，可以不用重写任务指令，可以把这些指令封装在一个新函数中，然后在必要时应用该函数就可以了。

R 函数是与前面看到的对象有类似结构的一种对象。作为对象，函数可以存储值。存储在函数中的“值”是一组指令，当 R 调用该函数时执行这组指令。因此，为了构建新函数，需要用赋值运算符把函数的内容存储到一个对象名（即函数名）中。

29 ≀ 30

下面举一个简单的例子。假设需要计算一组数据的均值的标准误差。根据定义，样本均值的标准误差为：

$$\text{标准误差} = \sqrt{\frac{s^2}{n}}$$

这里 s^2 是样本方差，n 是样本大小。

给定一个数值向量，现在需要计算相应的标准误差。先命名该函数为 se，在创建该函数之前，我们需要检查 R 中是否已经有一个同名的函数存在。如果的确有同名的函数存在，那么最好选择一个不同的名称；否则，会把 R 中已经存在的 R 函数对用户隐藏起来[⊖]。为了检查一个函

⊖ 这里对用户隐藏的 R 标准函数仍然是存在的。但是，在搜索路径中与它具有相同名称的用户自定义函数位于其顶部，因此“隐藏”了 R 的标准函数。

数在R中是否已经存在，在R命令行提示符处输入函数名就可以了：

```
> se

Error: Object "se" not found
```

R给出错误信息说明这个名称在R中不存在，可以放心使用。如果一个名称为se的函数（或者其他对象）已经存在，那么R将输出该对象的内容，而不是输出错误信息。

下面是创建新函数的一种可能方式：

```
> se <- function(x) {
+   v <- var(x)
+   n <- length(x)
+   return(sqrt(v/n))
+ }
```

因此，为了构建一个函数对象，使用下面的通用形式给函数名赋值：

```
function(<set of parameters>) { <set of R instructions> }
```

创建完成该函数后，可以用下列方式进行调用：

```
> se(c(45,2,3,5,76,2,4))

[1] 11.10310
```

与上面的se()函数一样，如果实现一个函数需要执行许多操作，那么需要以某种形式告诉R该函数体何时开始，何时结束。R用大括号作为一组指令的开始和结束的语法元素。

任何函数的返回值都可以由函数return()确定，或者R返回函数中最后运算表达式的结果。下面的函数演示了这一点，并应用了有默认值的参数。 31

```
> basic.stats <- function(x,more=F) {
+   stats <- list()
+
+   clean.x <- x[!is.na(x)]
+
+   stats$n <- length(x)
+   stats$nNAs <- stats$n-length(clean.x)
+
+   stats$mean <- mean(clean.x)
+   stats$std <- sd(clean.x)
+   stats$med <- median(clean.x)
+   if (more) {
+     stats$skew <- sum(((clean.x-stats$mean)/stats$std)^3) /
+                   length(clean.x)
+     stats$kurt <- sum(((clean.x-stats$mean)/stats$std)^4) /
+                   length(clean.x) - 3
+   }
+   unlist(stats)
+ }
```

上述函数的一个参数（或多个）有默认值（F）。这意味着调用该函数时可以设置或者不设置该参数。如果调用该函数时没有设定第二个参数的值，那么该参数将应用它的默认值。下例给出了这两种方式的示例：

```
> basic.stats(c(45, 2, 4, 46, 43, 65, NA, 6, -213, -3, -45))

       n     nNAs     mean      std      med
11.00000  1.00000 -5.00000 79.87768  5.00000
```

```
> basic.stats(c(45, 2, 4, 46, 43, 65, NA, 6, -213, -3, -45),
+     more = T)

          n      nNAs      mean       std       med      skew      kurt
11.000000  1.000000 -5.000000 79.877684  5.000000 -1.638217  1.708149
```

函数 basic. stats( ) 引入 R 的一种新指令：if( ) 指令。该指令可以使某些指令在逻辑测试为真值时执行。在上面的例子中，有两行指令分别计算向量值的偏度（skew）和峰度（kurt），如果变量 more 的值为真时，它们才执行；否则，它们将被跳过而不执行。

另一个重要的指令是 for( )。该指令允许一组指令重复地执行多次。下面是应用该指令的例子：

```
> f <- function(x) {
+   for(i in 1:10) {
+     res <- x*i
+     cat(x,'*',i,'=',res,'\n')
+   }
+ }
```

试着用某些数调用函数 f( ) （例如 f(5)）。上面的函数中的 for 指令告诉 R，它后面大括号内的指令需要执行多次（这里为 10 次）。也就是说，这些指令随着每次 *i* 取不同的值而被重复执行，*i* 的取值为集合 1：10，即 1，2，3，…，10。它意味着在 for 里的指令将被执行 10 次，每次 *i* 取不同的值。关键词 in 后面的值可以是任何向量，它们不必须是序列或者数值型。函数 cat( ) 用于将多个对象的内容输出到屏幕。也就是说，字符型数据将输出它自身（试运行 cat（'hello!'）），而其他对象将输出它们的内容（试运行 y <- 45，然后 cat(y)）。字符串 "\n" 使 R 换到下一行输出。

1.2.12 对象、类和方法

R 的设计原则之一是方便数据的操作，这样我们就可以容易地进行数据分析任务。在 R 中，数据以对象的形式存储。前面提到，R 中的一切，从简单的数值到函数、或更复杂的数据结构等，都是对象。每一个 R 对象都是属于一类。类定义了属于它们对象的抽象特征。也就是说，类定义了属于它们对象的特征和行为（或者方法）。例如，矩阵类（matrix 类）有维度属性和针对某些运算的一些特殊行为。事实上，当查询 R 的矩阵内容时，R 在屏幕上以一种特定的形式输出矩阵。这是因为所有来自矩阵类的对象都有一个相关联的特定输出方法。总之，一个对象所属的类将决定：1）某些泛型函数应用于这些对象时将应用的方法；2）对象的表示方法。对象的表示由存储在类对象的信息构成。

R 有许多预定义的对象类和相关的方法。基于这些已有的对象类，可以创建新的对象类和新方法。新方法可以是新类的方法，也可以是已有类的方法。新类一般建立在已有类的基础上，通常它们在已有类的表示上添加新信息。

类的表示由一组格子构成。每个格子有一个名称和决定它所存储信息相关联的类。运算符 "@" 用来获取存储在对象格子中的信息。$x@y$ 表示存储在对象 x 中的格子 y 的值。显然，它假定 x 所属的类含有一个名为 y 的格子的信息。

32 ~ 33

另一个与类有关的概念是类继承。继承建立了两个类之间的关系，它允许在一个已有类上添加额外的信息从而扩展为一个新类。这种类扩展意味着新类继承了已有类的所有方法，这将大大方便新类的创建，而不必从头开始。这时，我们仅仅需要实现新类中不同于它所扩展类的操作。

最后一个重要概念是多态。有些函数可以应用于多个不同的对象类，产生相应于该对象类的结果。在 R 中，多态与泛型函数紧密联系在一起。泛型函数实现了某些通用的高级操作。例

如，前面用到的函数 plot( ) 可以产生对象的图形表示，这是该函数的通用目的。然而，随着对象类的不同，它们的图形表示实际上是不同的。例如，绘制一组数值和绘制线性模型是不同的。用户只无需担忧这点，多态性是解决这种不同的关键。用户只需要知道有函数会产生对象的图形表示。R 和它的内部机制负责调度这些提供图形表示的特定类的通用函数。所有这些函数调度的细节都隐藏在 R 内部，用户无需了解这些琐碎的细节。事实上，R 知道 plot( ) 是一个泛型函数，它搜索适用于调用 plot( ) 函数的类对象的特定 plot 方法。如果存在这样一个特定的方法，那么 R 将应用该方法；否则，R 将调用默认的绘图方法。当用户决定创建一个新的对象类时，他们决定是否需要该对象类的特定方法。因此，当他们需要绘制新的类对象时，需要提供该对象类的特定绘图方法，该绘图方法将“告诉”R 如何绘制这个新的对象类。

以上是 R 的类和方法的基本概念。创建新类和相应的方法超出本书的讨论范围，读者可以参考有关 R 编程的书籍，例如 Chambers 的优秀书籍《Software for Data Analysis》(2008)。

1.2.13 管理 R 会话

当用 R 来解决复杂的问题时，交互式的命令行方式就变得很不方便。这时，实际的做法是把 R 代码写入一个文本文件中，然后让 R 来执行这些代码。为了生成这样的文件，可以用你喜欢的文本编辑器来编辑（例如记事本、Emacs 等），或者在 Windows 操作系统中，可以使用 R 自带的脚本编辑器，它在 R 软件的“文件”菜单中可以找到。当建立了代码文件并保存之后，可以在 R 命令行中用下面的命令来执行文件中的所有命令：

34

```
> source('mycode.R')
```

假定在当前工作目录中有一个名称为“mycode. R”[⊖]的文本文件。在 Windows 下，改变当前工作目录的简单方法是通过“文件”菜单中的“改变工作目录”选项。在 UNIX 操作系统中，可以分别用函数 getwd( ) 和 setwd( ) 来获取当前工作目录和改变当前工作目录。

在交互方式下应用 R 的命令行方式时，可能需要保存你的对象以备以后应用（例如，输入的函数）。下面的例子在名称为 mysession. RData 的文件中保存了名为 f 和 my. dataset 的两个对象：

```
> save(f,my.dataset,file='mysession.RData')
```

以后，在新的 R 会话中，可以使用下面的命令来重新读入这些对象：

```
> load('mysession.RData')
```

也可以用下面的命令来保存当前 R 工作空间中的所有对象[⊜]：

```
> save.image()
```

上面命令在当前工作目录中把 R 工作空间保存为名为“. RData”的文件。从该目录启动 R 时，这个文件将自动地载入 R 中。当 R 退出时，如果回答“是”，也会达到上面相同的效果（参见 1.2.1 节）。

关于 R 的参考文献

所有版本的 R 在线手册《R 简介》是学习 R 语言的极好资料。R 网站的“文档”部分的“贡献文档”也有关于 R 不同方面的免费书籍。

1.3 MySQL 简介

本节简单介绍 MySQL 数据库。在本书的案例研究中不一定要应用 MySQL。然而，对于大型

⊖ 这里的文件名后缀“. R”不是强制的，你可以用其他后缀名。

⊜ 前面提到过，可以用函数 ls( ) 来列出当前工作空间中的所有对象。

的数据挖掘项目，应用像 MySQL 这样的数据库是必需的。可以从网站 http://www.mysql.com 免费下载 MySQL 数据库。与 R 类似，MySQL 适用于多种不同的操作系统，例如 Linux、Windows 等。如果希望在计算机上安装 MySQL 数据库，应该首先从 MySQL 网站下载它，然后按照安装指南进行安装。另外，也可以访问安装在与你的计算机联网的其他计算机上的 MySQL 服务器。

35

可以应用网络上或者本地计算机上的客户端程序来访问 MySQL。在 MySQL 网站上，有多种不同的 MySQL 客户端程序。MySQL 自带一个控制台型的客户端程序，该控制台型客户端和 R 类似，以命令行方式工作。另外，也可以安装并应用图形界面的客户端程序来应用 MySQL，其中之一是 MySQL Query Browser，它是一款免费的 MySQL 客户端程序，可以试试。

如果应用控制台型的客户端程序，那么可以在操作系统的命令提示符后用以下命令来访问 MySQL：

```
$> mysql -u myuser -p

Password: ********

mysql>
```

如果访问远程的 MySQL 服务器，则应用下列命令：

```
$> mysql -h myserver.xpto.pt -u myuser -p

Password: ********

mysql>
```

这里假设 MySQL 服务器有一个名为“myuser”的用户并且有密码保护。如果上述内容对你而言很陌生，你应该与系统管理员进行交流或者学习 MySQL 安装手册，或者阅读相关书籍（例如，DuBois，2000）。

进入 MySQL 数据库后，可以应用已有的数据库或者创建新的数据库。下面说明如何在控制台型客户端用命令方式来创建一个新的数据库：

```
mysql> create database contacts;

Query OK, 1 row affected (0.09 sec)
```

可以通过以下命令来应用这个新建的数据库或者其他已经存在的数据库：

```
mysql> use contacts;

Database changed
```

数据库由一系列表构成，这些表包含相关条目的数据。创建表的方式如下：

```
mysql> create table people(
    -> id INT primary key,
    -> name CHAR(30),
    -> address CHAR(60));
Query OK, 1 row affected (0.09 sec)
```

注意符号“->”是表示继续 MySQL 命令的提示符。

为了在表中加入数据，一种方法是手动插入数据；另一种方法是用 MySQL 的导入命令读入外部文件中的数据，例如外部文本文件中的数据。

把一条记录插入表中的方法如下：

```
mysql> insert into people
    -> values(1,'John Smith','Strange Street, 34, Unknown City');

Query OK, 1 row affected (0.35 sec)
```

可以用 select 语句列出表中的记录。例如：

```
mysql> select * from people;

+----+------------+----------------------------------+
| id | name       | address                          |
+----+------------+----------------------------------+
|  1 | John Smith | Strange Street, 34, Unknown City |
+----+------------+----------------------------------+
1 row in set (0.04 sec)

mysql> select name, address from people;

+------------+----------------------------------+
| name       | address                          |
+------------+----------------------------------+
| John Smith | Strange Street, 34, Unknown City |
+------------+----------------------------------+
1 row in set (0.00 sec)

mysql> select name from people where id >= 1 and id < 10;

+------------+
| name       |
+------------+
| John Smith |
+------------+
1 row in set (0.00 sec)
```

当完成 MySQL 中的工作之后，可以在控制台型客户端用命令“quit”退出 MySQL。

36 ~ 37

关于 MySQL 的参考文献

可以阅读 MySQL 所带的免费手册来了解更多有关 MySQL 的知识。这个手册包含了从安装到 MySQL 所应用的 SQL 语言技术细节的所有内容。另外，DuBois（2000 年）所著的《MySQL》一书，也是学习 DBMS 的最佳参考资料，作者本身是一个活跃的 MySQL 开发人员。



预测海藻数量

本案例研究将介绍一些数据挖掘的基本任务：数据预处理、探索性数据分析和预测模型的构建。对于第一个案例研究，我们选择了一个就数据挖掘标准而言较小的问题。这个问题是预测水样中某几种有害海藻的存在频率。如果你不熟悉R语言，并且没有阅读1.2节，那么学习本章的案例时可能需要复习1.2节。

2.1 问题描述与目标

某些高浓度的有害藻类对河流生态环境的强大破坏是一个严重的问题，它们不仅破坏河流的生物，也破坏水质。能够监测并在早期对海藻的繁殖进行预测对提高河流质量是很有必要的。

针对这一问题的预测目标，在大约一年的时间内，在不同时间内收集了欧洲多条不同河流的水样。对于每个水样，测定了它们的不同化学性质以及7种有害藻类的存在频率。在水样收集过程中，也记录了一些其他特性，如收集的季节、河流大小和水流的速度。

本案例研究的主要动机之一是化学监测价格便宜，并且易于自动化。而通过分析生物样品来识别水中的藻类要涉及显微镜检验，需要训练有素的工作人员，因此既昂贵又缓慢。因此，构建一个可以基于化学性质来准确预测藻类的模型将有助于建立监测有害藻类的廉价的自动化系统。

本案例研究的另一个目的是更好地了解影响藻类频率的因素。也就是说，我们要了解藻类的频率和水样的某些化学性质以及其他特性（如季节、河流类型等）是如何相关的。

2.2 数据说明

本案例的数据来自于ERUDIT[1]研究网络，并被用于1999年的COIL国际数据分析竞赛。可以从多个不同的地方得到本案例的数据，例如UCI机器学习数据库[2]。

本案例有两个数据集，第一个数据集有200个水样。更精确地说，该数据集的每一条记录是同一条河流在该年的同一个季节的三个月内收集的水样的平均值。

每条记录由11个变量构成。其中3个变量是名义变量，它们分别描述水样收集的季节、收集样品的河流大小和河水速度。余下的8个变量是所观测水样的不同化学参数，即

- 最大pH值
- 最小含氧量（O_2）
- 平均氯化物含量（Cl）
- 平均硝酸盐含量（NO_3^-）
- 平均氨含量（NH_4^+）

---

[1] http://www.erudit.de/erudit/.

[2] http://archive.ics.uci.edu/ml/.

- 平均正磷酸盐含量（PO_4^{3-}）
- 平均磷酸盐含量（PO_4）
- 平均叶绿素含量

与这些参数相关的是 7 种不同有害藻类在相应水样中的频率数目。并未提供所观察藻类的名称的有关信息。

第二个数据集由 140 个额外观测值构成。它们的基本结构和第一个数据集一样，但是它不包含 7 种藻类的频率数目。这些额外的观测值可以视为测试集。本案例的主要目标是预测 140 个水样中 7 种藻类的频率。这意味着它是一个预测型数据挖掘任务。这是数据挖掘所处理的诸多问题中的一类问题。在这种问题中，任务是建立预测模型，并预测在给定预测变量的取值时相应的目标变量的值。预测模型也可能会说明哪一个预测变量对目标变量有较大的影响，即模型可能提供影响目标变量因素的一个综合描述。 40

2.3 数据加载到 R

我们考虑两种把数据载入 R 的方法：1）利用本书提供的 R 添加包，该包中给出了数据框形式的数据，可以直接应用；2）访问本书的网站，下载相应数据的文本文件，然后把文件读入到 R 中。很明显，第一种方式更实用。为了说明如何把文本文件载入 R 中，我们同时也介绍第二种方法。

如果应用第一种简单的方法，只要载入本书的 R 添加包[⊖]，就直接有了一个名为 algae 的数据框。这个数据框含有前面提到的 200 个观测值：

```
> library(DMwR)
> head(algae)

  season  size  speed mxPH mnO2     Cl    NO3     NH4    oPO4     PO4 Chla
1 winter small medium 8.00  9.8 60.800  6.238 578.000 105.000 170.000 50.0
2 spring small medium 8.35  8.0 57.750  1.288 370.000 428.750 558.750  1.3
3 autumn small medium 8.10 11.4 40.020  5.330 346.667 125.667 187.057 15.6
4 spring small medium 8.07  4.8 77.364  2.302  98.182  61.182 138.700  1.4
5 autumn small medium 8.06  9.0 55.350 10.416 233.700  58.222  97.580 10.5
6 winter small   high 8.25 13.1 65.750  9.248 430.000  18.250  56.667 28.4
    a1   a2   a3  a4   a5   a6  a7
1  0.0  0.0  0.0 0.0 34.2  8.3 0.0
2  1.4  7.6  4.8 1.9  6.7  0.0 2.1
3  3.3 53.6  1.9 0.0  0.0  0.0 9.7
4  3.1 41.0 18.9 0.0  1.4  0.0 1.4
5  9.2  2.9  7.5 0.0  7.5  4.1 1.0
6 15.1 14.6  1.4 0.0 22.5 12.6 2.9
```

数据框可以看做一种有列名称的矩阵或者表格，它是存储 R 数据表的一种理想的数据结构。 41 函数 head( ) 将显示数据框的前 6 行。

另外一种载入数据的方式是用本书网站中“Data”（数据）部分的文本文件。在“Training data”（训练数据）链接下的文件“Analysis. txt”中包含 200 个水样，而在“Test data”（测试数据）链接下的文件“Eval. txt”中包含 140 个测试样本。另一个链接下的文件“Sols. txt”包含 140 个测试样本的藻类频率。最后这个文件将用来测试预测模型的性能，它在建立模型时将被视为未知的。这些文件的每一行代表一个观测值。在训练集和测试集中，每一行的变量（如 2.2 节所描述）值之间由空格来分隔。缺失值由字符串“XXXXXXX”来表示。

⊖ 由于该添加包在 R 的标准安装中是没有的，必须先安装它才能使用。关于添加包的安装，请参见 1.2.1 节。

我们需要做的第一件事是从本书网站下载三个文件，并把它们存储在本地计算机硬盘的某个目录下（最好存储在当前运行 R 的目录下，可以在 R 命令行下用命令 getwd() 来获取该目录）。

把文件下载到一个本地计算机目录后，就可以把文本文件"Analysis.txt"中的数据加载到 R 中（训练数据是指用来建立预测模型的数据）。可以输入以下命令把文件中的数据读入到 R 中[㊀]：

```
> algae <- read.table('Analysis.txt',
+           header=F,
+           dec='.',
+           col.names=c('season','size','speed','mxPH','mnO2','Cl',
+           'NO3','NH4','oPO4','PO4','Chla','a1','a2','a3','a4',
+           'a5','a6','a7'),
+           na.strings=c('XXXXXXX'))
```

参数 header = F 表示要读的文件的第一行不包括变量名。dec = '.' 指出数值使用'.'字符分隔小数位。这两个参数设置可以省略，因为这里使用了它们的默认值。col. names 给正在读取的变量提供一个名称向量。最后，na. strings 表示该字符串将被解释为未知值。这些值在 R 内部用 NA 来表示，如 1.2.3 节中所示。

R 有多个读取文本文件中数据的函数。可以输入"? read. table"来获得进一步的信息和其他相关函数。此外，R 有一本名为《R Data Import/Export》的手册，该手册描述了 R 从其他应用程序读取数据的不同方法。

42

上面指令的结果是一个数据框。数据框的每一行代表数据集的一个观测值。例如，可以用指令 algae [1：5,][㊁]来获得文件的前 5 个观测值。在 1.2.7 节我们讲了在 R 中获取像数据框这样的 R 对象的特定元素的方法。

2.4 数据可视化和摘要

鉴于没有该问题领域足够的信息，首先了解一些数据的统计特性是一种较好的方式，它方便我们更好地理解问题。即使对问题有了充分的了解，从下面的探索性数据分析着手进行分析也是一个不错的方法。

获取数据统计特性的一个方法是获取数据的如下描述性统计摘要。

```
> summary(algae)

   season       size        speed         mxPH            mnO2
 autumn:40   large :45   high  :84   Min.   :5.600   Min.   : 1.500
 spring:53   medium:84   low   :33   1st Qu.:7.700   1st Qu.: 7.725
 summer:45   small :71   medium:83   Median :8.060   Median : 9.800
 winter:62                           Mean   :8.012   Mean   : 9.118
                                     3rd Qu.:8.400   3rd Qu.:10.800
                                     Max.   :9.700   Max.   :13.400
                                     NA's   :1.000   NA's   : 2.000
       Cl                NO3                NH4                oPO4
 Min.   :  0.222   Min.   : 0.050   Min.   :    5.00   Min.   :  1.00
 1st Qu.: 10.981   1st Qu.: 1.296   1st Qu.:   38.33   1st Qu.: 15.70
 Median : 32.730   Median : 2.675   Median :  103.17   Median : 40.15
 Mean   : 43.636   Mean   : 3.282   Mean   :  501.30   Mean   : 73.59
 3rd Qu.: 57.824   3rd Qu.: 4.446   3rd Qu.:  226.95   3rd Qu.: 99.33
```

㊀ 这里假设数据文件位于 R 的当前工作目录中；否则，可以应用函数 setwd() 来改变当前工作目录。在 Windows 操作系统中，可以点击"文件"菜单下的"改变工作目录"选项来改变 R 的当前工作目录。

㊁ 这里也可以用前面介绍的函数 head (algae) 得到相同的结果。

```
 Max.   :391.500   Max.   :45.650   Max.   :24064.00   Max.   :564.60
 NA's   : 10.000   NA's   : 2.000   NA's   :    2.00   NA's   :  2.00
      PO4              Chla               a1                a2
 Min.   :  1.00   Min.   :  0.200   Min.   : 0.00   Min.   : 0.000
 1st Qu.: 41.38   1st Qu.:  2.000   1st Qu.: 1.50   1st Qu.: 0.000
 Median :103.29   Median :  5.475   Median : 6.95   Median : 3.000
 Mean   :137.88   Mean   : 13.971   Mean   :16.92   Mean   : 7.458
 3rd Qu.:213.75   3rd Qu.: 18.308   3rd Qu.:24.80   3rd Qu.:11.375
 Max.   :771.60   Max.   :110.456   Max.   :89.80   Max.   :72.600
 NA's   :  2.00   NA's   : 12.000
       a3                a4                a5                a6
 Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
 1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.000
 Median : 1.550   Median : 0.000   Median : 1.900   Median : 0.000
 Mean   : 4.309   Mean   : 1.992   Mean   : 5.064   Mean   : 5.964
 3rd Qu.: 4.925   3rd Qu.: 2.400   3rd Qu.: 7.500   3rd Qu.: 6.925
 Max.   :42.800   Max.   :44.600   Max.   :44.400   Max.   :77.600

       a7
 Min.   : 0.000
 1st Qu.: 0.000
 Median : 1.000
 Mean   : 2.495
 3rd Qu.: 2.400
 Max.   :31.600
```

这个简单的指令立即给出了数据的统计特性概览[㊀]。对于名义变量（R 中用因子来表示），它给出每个可能取值的频数[㊁]。例如，从结果中可知冬季采集的水样比其他季节更多。对于数值变量，R 为我们提供了均值、中位数、四分位数及极值等一系列的统计信息。这些统计信息提供了变量值分布的初步信息（后面还会有这方面的分析）。在变量有缺失值的情况下，字符串 NA 后面的数字即为缺失值的个数。通过观察中位数和均值之间的差异以及四分位距[㊂]，我们可以了解数据分布的偏度和分散情况。另外，大部分情况下，这些信息可以更好地用图形来表示出来。让我们看一个例子。

```
> hist(algae$mxPH, prob = T)
```

该指令将绘制变量 mxPH 的直方图。其结果如图 2-1 所示。设置参数 prob = T，我们可以得到每个取值区间[㊃]的概率，如果该参数设置为 FALSE 或者忽略该参数，它将给出频数。

图 2-1 告诉我们，变量 mxPH 的分布非常接近正态分布，它的值大部分聚集在该变量的均值周围。我们通过使用 Q－Q 图来检验该变量是否为正态分布。在 R 的添加包 car（Fox，2009）中的函数 qq. plot( ) 可以绘制 Q－Q 图。上例的 Q－Q 图如图 2-2 的右图所示，左图是一个略微复杂版本的直方图。获取图 2-2 的命令如下：

㊀ 应用添加包 Hmisc 中的函数 describe( ) 也可以得到类似的结果（Harrell Jr，2009）。

㊁ 事实上，如果有太多的取值，则只显示出现频次最高的几个取值。

㊂ 如果把变量的取值按照从小到大的顺序排列，25% 的取值小于第一个四分位数，而 75% 的数值小于第三个四分位数，因此在第一个四分位数和第三个四分位数之间的数值个数为 50%。四分位距是第三个四分位数和第一个四分位数的差值，它可以衡量变量与其中心值的偏离程度，该值越大说明偏离越大。

㊃ 直方图的条形的面积之和应该为 1（而不是某些人认为的条形的高度）。

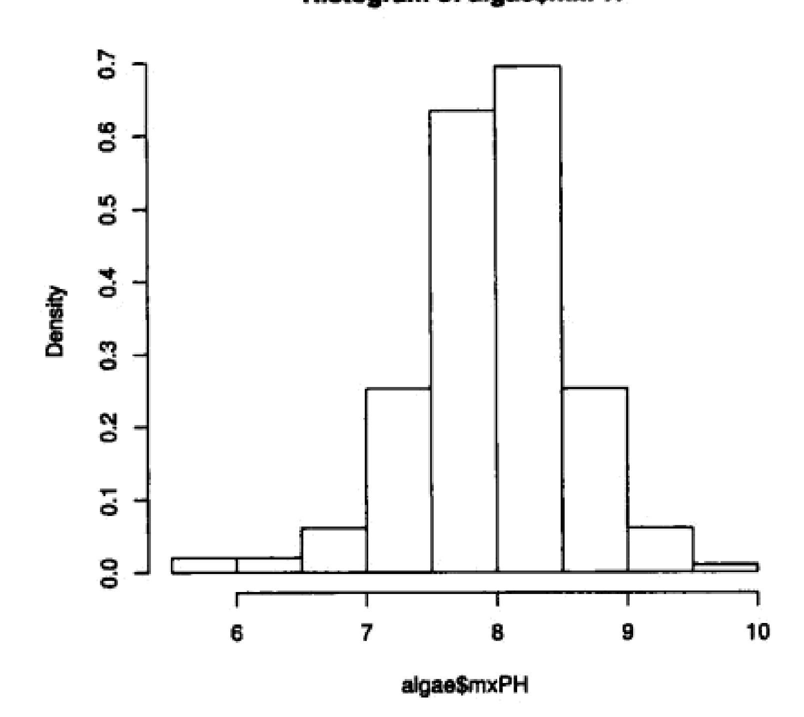


图 2-1　变量 mxPH 的直方图

```
> library(car)
> par(mfrow=c(1,2))
> hist(algae$mxPH, prob=T, xlab='',
+       main='Histogram of maximum pH value',ylim=0:1)
> lines(density(algae$mxPH,na.rm=T))
> rug(jitter(algae$mxPH))
> qq.plot(algae$mxPH,main='Normal QQ plot of maximum pH')
> par(mfrow=c(1,1))
```

载入 R 添加包 car 后[㊀]，调用函数 par( ) 设置 R 图形系统的多个参数。这里把图形输出窗口设置为 1 行 2 列的区域，这样可以在同一幅图形中得到两个并列的图形。然后绘制第一幅图形，它是变量 mxPH 的直方图，这里它设置 *X* 轴标题为空，然后改变图形的标题，提供合理的 *Y* 轴的范围。之后的指令绘制平滑版本的直方图（变量分布的核密度估计[㊁]），而下一个命令在 *X* 轴附近绘制变量的实际值，从而容易识别离群点[㊂]。例如，我们可以观察到有两个值显著低于所有其他值。这种数据检查是非常重要的，因为它可以确定数据样本中可能出现的错误，甚至帮助定位那些奇怪的错误值或者在后续分析中需要剔除的奇怪值。图 2-2 的右图是用函数 qq. plot( ) 得到的 Q－Q 图，它绘制变量值和正态分布的理论分位数（黑色实线）的散点图。同时，它给出正态分布的 95% 置信区间的带状图（虚线）。从图 2-2 右图可知，变量有几个小的值明显在 95% 置信区间之外，它们不服从正态分布。

43 ~ 45

注意，上例大量应用了函数复合，一个函数的结果调用另一个函数。当不理解这些函数复合时，可以每次调用一个函数，分别理解它的输出。

下面的指令给出了另一个数据检查的例子，它用来检查变量 oPO4：

㊀ 这里应用函数 library( ) 来载入添加包时要注意，该添加包必须预先安装在你的计算机中。否则，R 会报错，这时，需要用 1.2.1 节中给出的方法来先安装该添加包。

㊁ 在许多函数中，设置参数 "na. rm = T" 是说明在函数的计算中不考虑 NA 值。在多个函数中，这种设置是必需的，因为它不是这些函数的默认设置。否则，将会出现错误。

㊂ 事实上，这里有两个函数调用。函数 rug( ) 执行绘图，而函数 jitter( ) 对要绘制的原始值略微进行随机排列，这就避免了两个值相等的可能性，因而避免了两个标记重合在一起而导致可视化检查时一些值被"掩盖"。

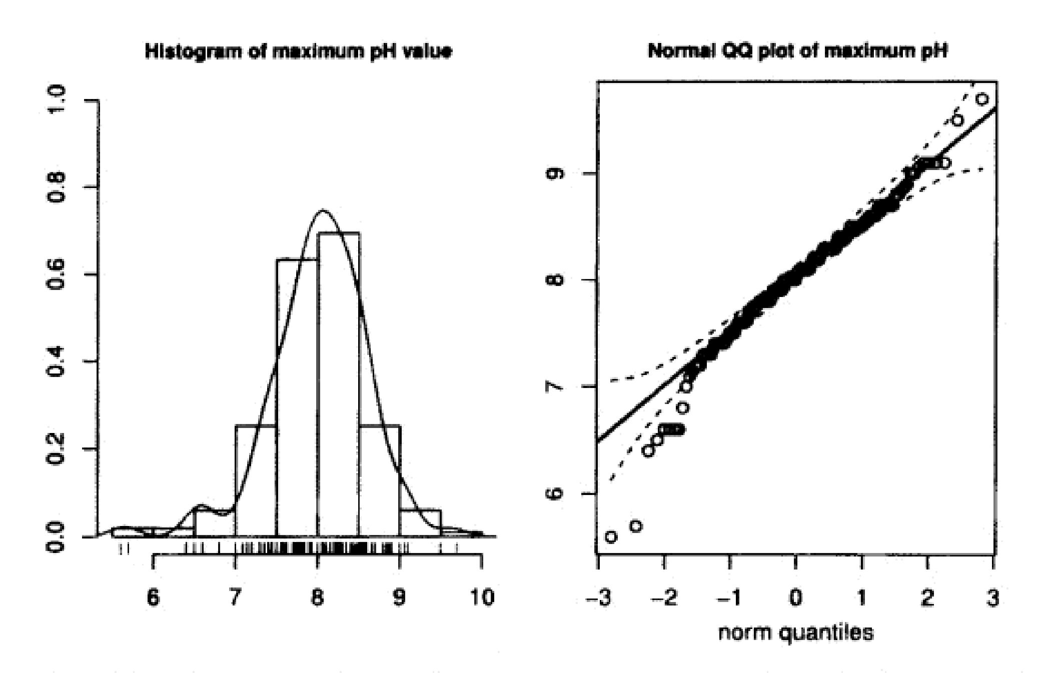


图 2-2 变量 MxPH 的直方图的“丰富”版本（左图）以及 Q－Q 图（右图）

```
> boxplot(algae$oPO4, ylab = "Orthophosphate (oPO4)")
> rug(jitter(algae$oPO4), side = 2)
> abline(h = mean(algae$oPO4, na.rm = T), lty = 2)
```

第一条指令绘制变量 oPO4 的箱图。箱图能快速提供变量分布的一些关键属性的摘要。箱图框的边界代表变量的第一个四分位数和第三个四分位数，而框内的水平线是变量的中位数。设 r 是变量的四分位距，箱图上方的小横线是小于或等于第三个四分位数加 $1.5 \times r$ 的最大的观测值，而箱图下方的小横线是大于或等于第一个四分位数减去 $1.5 \times r$ 的最小的观测值。箱图上方小横线上面或者下方小横线下面的小圆圈表示与其他值相比特别大或者特别小的值，通常认为是离群值。这意味着箱图给出大量的信息，它不仅给出了变量的中心趋势，也给出了变量的发散情况和离群值。

46 ~ 47

第二条指令在前面已经描述过了（唯一的区别是数据绘制的位置不同），而第三条指令使用 abline( ) 在变量的均值位置绘制一条水平线[⊖]，均值由函数 mean( ) 计算。将均值线和箱图内的分位数线进行比较，就可以知道变量的多个离群值使得作为变量中心（即变量的大部分取值）的均值产生了扭曲。

图 2-3 的分析说明，变量 oPO4 的分布集中在较小的观测值周围，因此分布为正偏。大部分水样的 oPO4 值比较低，但也有几个水样的观测值较高，甚至特别高。

有时，当有离群值时，需要确定那些有离群值的观测。这里给出两种方法。一种方法是图形方法。如果绘制变量 NH4 的值，将会注意到一个特别大的值。我们可以用下列方式识别特大值相应的水样：

```
> plot(algae$NH4, xlab = "")
> abline(h = mean(algae$NH4, na.rm = T), lty = 1)
> abline(h = mean(algae$NH4, na.rm = T) + sd(algae$NH4, na.rm = T),
+     lty = 2)
> abline(h = median(algae$NH4, na.rm = T), lty = 3)
> identify(algae$NH4)
```

第一条指令绘制变量的所有值，调用函数 abline( ) 绘制三条有用的直线：第一条为均值，第二条为均值加 1 个标准差，第三条为中位数。对于离群值的识别，尽管这三条线不是必需的，



⊖ 用参数 lty = 2 来设置线型为虚线。

但是它们能提供变量的有用信息。最后一条指令是交互式的，它允许用户单击图形中的点[⊖]。对于每一个单击的点，R 将写下该点在 algae 数据框中的行号。用户可以右击来结束交互。

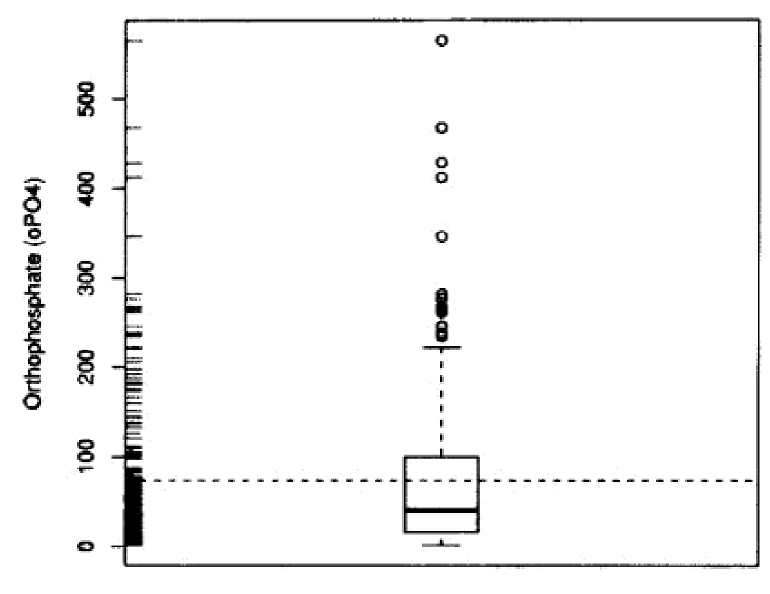


图 2-3 变量 oPO4 箱图的“丰富”版本

如果需要检查 algae 数据框中对应于图形中的离群值的观测记录，最好的方式是执行如下命令：

```
> plot(algae$NH4, xlab = "")
> clicked.lines <- identify(algae$NH4)
> algae[clicked.lines, ]
```

正如你所猜测的那样，函数 identify( ) 给出对应于图形中单击的点的行号，利用这点对 algae 数据框进行索引，可以获取这些观测值的所有信息。

48

也可以不用图形方式进行以上的检查，如下所示：

```
> algae[algae$NH4 > 19000, ]
```

该指令给出了另一种索引数据框的方式，即应用逻辑表达式来选定行（更多的例子请参见 1.2.7 节）。这条指令的输出结果可能看起来有点奇怪。原因是变量 NH4 的有些观测值为 NA 值，这时 R 不知道比较的结果，因此给出了 NA。为了避免这种情况，可以应用指令 algae[!is.na(algae$NH4) & algae$NH4 > 19000,]。调用函数 is.na( ) 将产生一个布尔值（TRUE 或者 FALSE）向量。当 NH4 的值为 NA 时，向量的相应值为 TRUE，这个向量元素的个数和数据框 algae 的行数相同。表达式!is.na（algae$NH4）返回一个布尔值向量，因为“!”是逻辑否运算，所以对应数据框中变量 NH4 的值为已知的行的位置的值为 TRUE。总之，这种索引方式将给出那些数据框中变量 NH4 取值已知并且大于 19 000 的行。

下面给出几种其他类型的数据检查的例子。这些例子应用 R 的添加包 lattice（Sarkar, 2010），lattice 包提供了大量优秀的图形工具，这些图形工具实现了 Trellis 图形（Cleveland, 1993）的思想。

假设需要研究海藻变量 *a*1 的值的分布。可以应用上面讨论的任何方法。然而，如果这里需要研究分布如何依赖于其他变量，就需要新的工具。

条件绘图是依赖于某个特定因子的图形表示。因子是一个取值为有限集合的名义变量。例如，对于变量 size 的不同取值，可以绘制变量 *a*1 的一组箱图（如图 2-4 所示）。每个箱图是对应于变量 size 的某个特定值的水样子集。通过这些箱图可以研究名义变量 size 如何影响变量 *a*1 值

⊖ 鼠标单击的相对于点的位置将决定 R 把行号写在哪一边。例如，如果在点的右边单击，行号将写在点的右边。

的分布。绘制图 2-4 中箱图的命令如下：

```
> library(lattice)
> bwplot(size ~ a1, data=algae, ylab='River Size',xlab='Algal A1')
```

上面的第一条指令载入 lattice 包。第二条指令绘制这些图 lattice 版本的箱图，这条指令可以读做：对变量 size 的每个值绘制 $a1$。其他参数的意义显而易见。

从图 2-4 可知，在规模较小的河流中，海藻 $a1$ 的频率较高，这是很有用的信息。

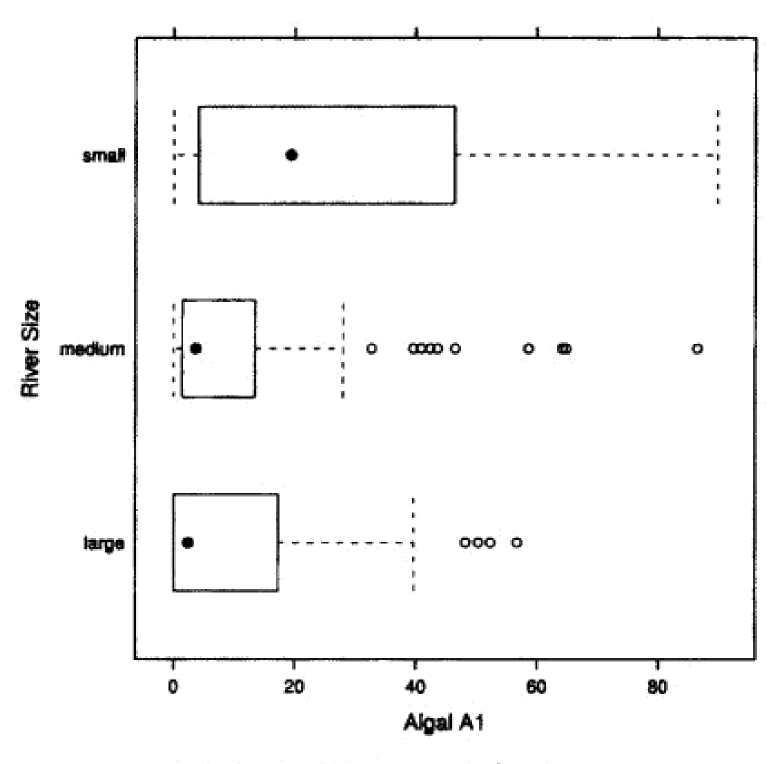


图 2-4　海藻变量 $a1$ 的条件箱图

这种箱图的另外一个类型是分位箱图，它可以给出所绘制变量的更多信息。R 添加包 Hmisc 可以绘制分位箱图。下面绘制上面例子中 $a1$ 变量的条件分位箱图：

49

```
> library(Hmisc)
> bwplot(size ~ a1, data=algae,panel=panel.bpplot,
+         probs=seq(.01,.49,by=.01), datadensity=TRUE,
+         ylab='River Size',xlab='Algal A1')
```

上面命令的输出结果如图 2-5 所示。图 2-5 中的点代表不同大小的河流中海藻频数的均值，而图中的竖线分别代表变量的第一个分位数、中位数和第三个分位数。图 2-5 中的小竖线代表数据的真实取值，这些值的分布信息则由分位数图来体现。分位数箱图提供的信息要多于图 2-4 所示的传统箱图的信息。例如，我们可以确认上面的观测结论：小型的河流有更高频率的海藻，但我们也观察到小型河流的海藻频率的分布比其他类型河流的海藻频率的分布分散。

这种类型的条件绘图不局限于名义变量，也不局限于单个因子。只要先把连续变量“离散化”，也同样可以进行条件绘图。下面给出一个两个因子的条件绘图的例子。考虑变量 $a3$ 在给定变量 season 和变量 mnO2 下的条件绘图，变量 mnO2 是一个连续变量，绘图代码如下所示：

```
> minO2 <- equal.count(na.omit(algae$mnO2),
+                      number=4,overlap=1/5)
> stripplot(season ~ a3|minO2,
+          data=algae[!is.na(algae$mnO2),])
```

以上代码得到的图形输出如图 2-6 所示。

上面代码的第一行是调用函数 equal. count( ) 对连续变量 mnO2 离散化，把该变量转换为因子类型。参数 number 设置需要的区间个数，参数 overlap 设置两个区间之间的靠近边界的重合

（这意味着某些观测值将被分配到相邻的区间中）。每个区间的观测值的个数相等。注意，变量 algae$mnO2 中含有 NA 值，所以上面的指令中没有直接应用该变量，否则会导致其后的绘图函数出错。函数 na. omit( ) 可以用来剔除向量中的任何 NA 值[⊖]。

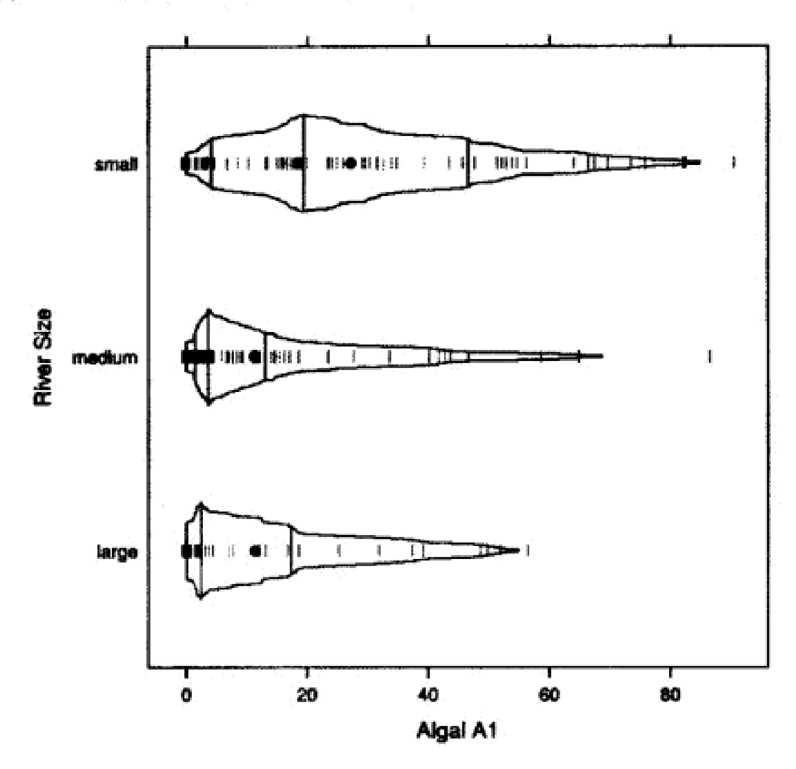


图 2-5　海藻变量 $a1$ 的条件分位数箱图

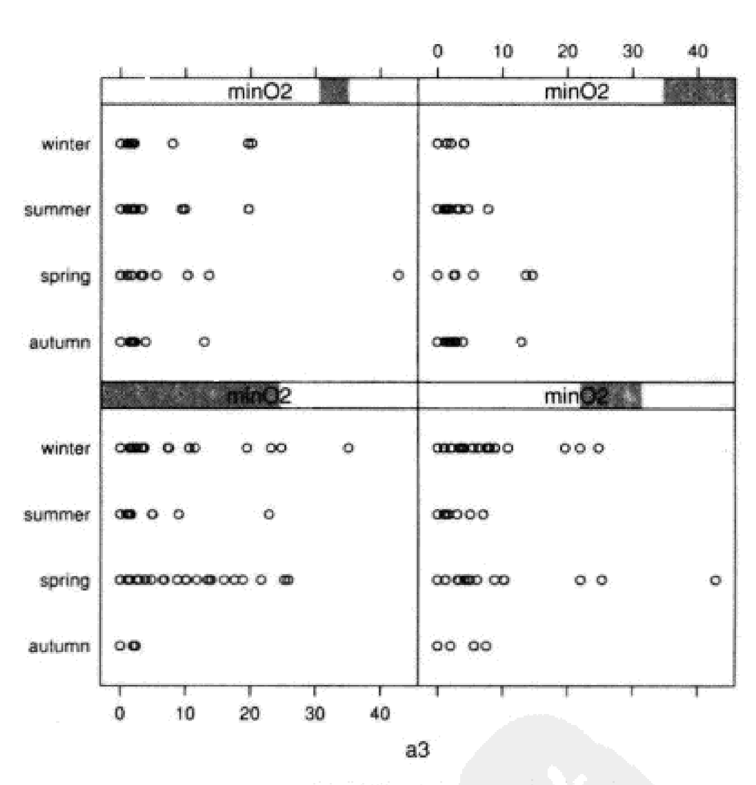


图 2-6　海藻变量 $a3$ 的条件箱图

第二行调用绘图函数 stripplot( )，该函数是 lattice 包中的一个绘图函数，它根据另一个变量（这里是 season）把变量的实际值绘制到不同的图形中。然后对变量 mnO2 的每个不同的区间绘制不同的图形。这些区间按照从左到右、从下到上的顺序来排列。即与左下方的图形相对应的是较小的 mnO2 值[⊜]。变量 mnO2 中的 NA 值也会对图形的绘制产生影响。不能像绘制图 2-4 那样直

50 ～ 51

⊖ 在后面的 2.5 节将给出另一种更好的解决方案。

⊜ 可以打印所创建的变量离散化版本来查看创建的区间的实际值。

接应用参数 data = algae，而应该先剔除水样中变量 mnO2 含有 NA 值的行。

数据摘要和可视化的参考文献

大多数标准的统计教科书都含有数据汇总的内容。一本较简单的、很好的统计书籍是 Chatfield（1983）的《Statistics for Technology》。该书的例子简单并很能说明问题。另外一本好的参考书籍是 Dalgaard（2002）的《Introductry Statistics with R》。对于数据可视化，必需的一本参考书是 Cleveland（1993）的《Visualizing Data》，这时一本物有所值的优秀书籍。另外一本后续的更正式的书籍是《The Elements of Graphing Data》（Cleveland，1995）。一本更新的优秀书籍是 Chen 等（2008）编著的《Data Visualization》。最后，Murrell（2006）的《R Graphics》则更是一本全部有关 R 软件绘图的书籍。

2.5 数据缺失

在许多水样中，一些变量含有缺失值。这种情形在现实问题中非常普遍，这会导致一些不能处理缺失值的分析方法无法应用。 52

当我们处理含有缺失值的数据时，可以运用以下几种最常见的策略：

- 将含有缺失值的案例剔除。
- 根据变量之间的相关关系填补缺失值。
- 根据案例之间的相似性填补缺失值。
- 使用能够处理缺失值数据的工具。

最后一种方式是最严格的，因为它限定了我们可以使用的工具。然而，当对这些可以处理缺失值的数据挖掘工具有信心时，它们可能是一个好的选择。在下面的章节中，我们将通过举例来演示如何在 R 中实现上述处理缺失值的方法。如果你决定试一试本书中给出的代码，你需要了解它们不是互相补充的。由于每节都用不同的方法来处理这些数据，这意味着当你选择用其他方式来处理这些缺失值时，需要重新读取含有所有缺失值的原始数据。重新读取数据的代码如下：

```
> library(DMwR)
> data(algae)
```

2.5.1 将缺失部分剔除

剔除含有缺失数据的记录非常容易实现，尤其是当这些记录所占的比例在可用数据集中非常小的时候，这个选择就比较合理。

在剔除某些变量中至少含有一个缺失数据的所有观测值时，最好先检查观测值，或者至少得到这些观测值的个数，例如：

```
> algae[!complete.cases(algae),]

...
...

> nrow(algae[!complete.cases(algae),])

[1] 16
```

函数 complete. cases( ) 产生一个布尔值向量，该向量的元素个数与 algae 数据框中的行数相同，如果数据框的相应行中不含 NA 值（即为一个完整的观测值），函数返回值就是 TURE。前面提到过“!”运算符，它是取逻辑否，因此上述指令显示了含有缺失值的水样记录。

为了从数据框中剔除这 16 个样本，我们可以简单地输入： 53

```
> algae <- na.omit(algae)
```

即使我们决定不使用剔除所有包含缺失值记录的极端方法，我们也可以剔除某些观测值。因为



这些样本的缺失值太多，所以它们几乎是无用的样本，如果采用复杂的方法来填补缺失值，就会导致较大偏差。需要注意的是，如果执行了前面的命令，你需要重新读取数据，因为这个指令已经剔除了所有缺失数据，所以接下来的命令就没有意义！观察这个样本中的数据，我们可以看到第62条和第199条记录中的11个解释变量有6个是缺失值。在这种情况下，最好是剔除它们：

```
> algae <- algae[-c(62, 199), ]
```

在有些问题中，由于大量记录中含有缺失值，用上面的观察方法来检查数据的缺失值是不可行的，所以需要找出缺失值较多的样本所在的行。下面的代码可以找出海藻数据集中每行数据的缺失值个数：

```
> apply(algae, 1, function(x) sum(is.na(x)))
```

函数 apply( ) 属于 R 中功能非常强大的一类函数。这类函数又称为元函数，它们可以在某些条件下对对象应用其他函数。对函数 apply( ) 而言，它可以把任何其他函数应用到一个多维对象的各个维度上。使用函数 apply( ) 时，它把一个函数应用到数据框[⊖]的每一行。这个被应用的函数在 apply( ) 函数的第三个参数中给出，对数据框的每一行都分别调用该函数。在这个案例中我们使用一个临时函数。它只在调用 apply( ) 函数时才存在。另外，函数 apply( ) 的第三个参数也可以是一个“正常”函数的函数名。临时函数的功能是计算对象 x 中 NA 的数量。在 R 中逻辑值 TURE 等于数值 1，逻辑值 FALSE 等于数值 O，这意味着当加一个布尔值向量时，得到向量中取值为 TURE 的元素的个数。

根据以上代码，可以编写一个程序找出 algae 中含有给定数目缺失值的行。在本书提供的添加包中有这个函数。可以如下应用该函数：

```
> data(algae)
> manyNAs(algae, 0.2)
[1]  62 199
```

54

只有在前面操作中剔除了缺失值较多的几行数据后才需要调用函数 data( )。函数 manyNAs( ) 的功能是找出缺失值个数大于列数 20% 的行。在第二个参数中可以设置一个精确的列数作为界限。因此，用下面的代码就无须知道含有缺失值较多的行的具体数量：

```
> algae <- algae[-manyNAs(algae), ]
```

在这个案例中我们应用了 manyNAs( )，函数第二个参数的默认值为 0.2。

2.5.2 用最高频率值来填补缺失值

填补含有缺失值记录的另一个方法是尝试找到这些缺失值最可能的值。同样，这里有多种策略可供选择，不同策略对逼近程度和算法复杂度的权衡不同。

填补缺失数据最简便和快捷的方法是使用一些代表中心趋势的值。代表中心趋势的值反映了变量分布的最常见值，因此中心趋势值是最自然的选择。有多个代表数据中心趋势的指标，例如平均值、中位数、众数等。最合适的选择由变量的分布决定。对于接近正态的分布来说，所有的观测值都较好地聚集在平均值周围，平均值数就是最佳选择。然而，对于偏态分布，或者有离群值的变量来说，选择平均值就不好。偏态分布的大部分值都聚集在变量分布的一侧，因此平均值不能作为最常见值的代表。另一方面，离群值（极值）的存在会扭曲平均值[⊜]，这就导致了平均值不具有代表性的问题。因此，在对变量分布进行检查之前选择平均值作为中心趋势的代表

⊖ 第二个参数中的“1”表示第一个参数中的对象的第一个维度，即数据框的行数据。

⊜ 向量 c(1.2, 1.3, 0.4, 0.6, 3, 15)的均值是3.583。



是不明智的，例如，某些R的绘图工具（见图2-2）。对偏态分布或者有离群值的分布而言，中位数是更好的代表数据中心趋势的指标。

比如，样本 algae[48,] 中的变量 mxPH 有缺失值。由于该变量分布近似正态分布（见图2-2），我们可以选用平均值来填补这个“洞”，计算方法如下：

```
> algae[48, "mxPH"] <- mean(algae$mxPH, na.rm = T)
```

55

这里，函数 mean( ) 计算数值向量的平均值，参数 na.rm = T 使计算时忽略缺失数据[⊖]。

大多数时候采用一次填补一列中的所有缺失值而不是像上面那样一行一行地逐个填补。以变量 Chla 为例，这个变量在第12行上有缺失值。另外，这也是平均值不能代表大多数变量值的一种情况。事实上，Chla 的分布偏向于较低的数值，并且它有几个极端值，这些都使得平均值（13.971）不能代表大多数的变量值。因此，我们使用中位数来填补这一类的缺失值：

```
> algae[is.na(algae$Chla), "Chla"] <- median(algae$Chla, na.rm = T)
```

本书插件包中提供的函数 centralImputation( ) 可以用数据的中心趋势值来填补数据集的所有缺失值。对数值型变量，该函数用中位数；对名义变量，它采用众数。该函数的应用如下：

```
> data(algae)
> algae <- algae[-manyNAs(algae), ]
> algae <- centralImputation(algae)
```

由于缺失值的存在会导致某些方法不能使用，所以使用上面的方法填补缺失值通常也认为不是很好的方法。虽然上述的简单方法速度快，特别适用于大数据集，但是它可能导致较大的数据偏差，影响后期的数据分析工作。然而，使用无偏方法来寻找最佳数据填补值复杂，对于大型数据挖掘问题可能并不适用。

2.5.3 通过变量的相关关系来填补缺失值

另一种获取缺失值较少偏差估计值的方法是探寻变量之间的相关关系。比如，通过变量值之间的相关关系，能够发现某变量与 mxPH 高度相关。这可以使我们得到含有缺失值的第48条样本的更可能的填补值。这比之前使用平均值的方法将更胜一筹。

应用如下命令来得到变量间的相关值：

```
> cor(algae[, 4:18], use = "complete.obs")
```

56

函数 cor( ) 的功能是产生变量之间的相关值矩阵（因为前3个变量是名义变量，所以计算相关值时不考虑它们）。设定参数 use = "complete.obs" 时，R 在计算相关值时忽略含有 NA 的记录。相关值在1（或-1）周围表示相应的两个变量之间有强正（或负）线性相关关系。然后其他 R 函数可以得到变量间线性相关的近似函数形式，它可以让我们通过一个变量的值计算出另一个变量的值。

函数 cor( ) 的输出结果并不是很清晰，但可以通过函数 symnum( ) 来改善结果的输出形式，例如：

```
> symnum(cor(algae[,4:18],use="complete.obs"))

     mP mO Cl NO NH o P Ch a1 a2 a3 a4 a5 a6 a7
mxPH 1
mnO2    1
Cl         1
NO3           1
```

⊖ 因为原始数据在该列中有缺失值，如果不设定参数 na.rm = T，得到的结果将是 NA。



```
NH4           , 1
oPO4    . .        1
PO4     . .        * 1
Chla .                 1
a1          .      . .   1
a2    .             .      1
a3                           1
a4       .        .            1
a5                               1
a6             . .             .   1
a7                                   1
attr(,"legend")
[1] 0 ' ' 0.3 '.' 0.6 ',' 0.8 '+' 0.9 '*' 0.95 'B' 1
```

这种用符号表示相关值的方法更为清晰，特别是对于大的相关矩阵。

在本案例的数据中，大多数变量之间是不相关的。然而，有两个例外：变量 NH4 和 NO3 之间，变量 PO4 和 oPO4 之间。后两个变量之间的相关值很高（大于0.9）。变量 NH4 和 NO3 之间的相关性不是特别明显（为0.72），因此根据它们来确定缺失数据是危险的。此外，因为样本62和样本 199 有太多的变量含有缺失值，所以如果剔除它们，样本中的变量 NH4 和 NO3 就没有缺失值了。至于变量 PO4 和 oPO4，它们之间相关性[⊖]可以帮助填补这两个变量的缺失值。为了达到这个目标，我们需要找到这两个变量之间的线性相关关系，方法如下：

57

```
> data(algae)
> algae <- algae[-manyNAs(algae), ]
> lm(PO4 ~ oPO4, data = algae)

Call:
lm(formula = PO4 ~ oPO4, data = algae)

Coefficients:
(Intercept)         oPO4
     42.897        1.293
```

函数 lm( ) 可以用来获取形如 $Y=\beta_0+\beta_1 x_1+\cdots+\beta_n x_n$ 的线性模型。2.6 节将具体讲述该函数。线性模型是：PO4 = 42.897 + 1.293 × oPO4。如果这两个变量不是同时有缺失值，那么可以通过这个公式计算这些变量的缺失值。

在剔除样本 62 和样本 199 后，还剩下一个样本（样本 28）在变量 PO4 上有缺失值，可以简单地使用上面的线性关系计算缺失值的填补值：

```
> algae[28, "PO4"] <- 42.897 + 1.293 * algae[28, "oPO4"]
```

然而，为了说明这个方法，我们假设变量 PO4 有多个缺失值。如何使用上述的线性关系计算所有的缺失值呢？最好的方法是构造一个函数，它可以根据给定的 oPO4 的值计算 PO4 的值，然后对所有缺失值应用这个函数。

```
> data(algae)
> algae <- algae[-manyNAs(algae), ]
> fillPO4 <- function(oP) {
+     if (is.na(oP))
+         return(NA)
```

⊖ 根据领域专家的解释，总的磷酸盐值（PO4）包含正磷酸盐值（oPO4）。

```
+      else return(42.897 + 1.293 * oP)
+ }
> algae[is.na(algae$PO4), "PO4"] <- sapply(algae[is.na(algae$PO4),
+      "oPO4"], fillPO4)
```

上面代码创建了一个叫做 fillPO4( ) 的函数，该函数有一个参数来接收变量 oPO4 的值。给定 oPO4 的值，这个函数将根据得到的线性关系（尝试“fillPO4(6.5)”语句）计算变量 PO4 的值。然后，将这个函数应用到变量 PO4 有缺失值的所有样本。这个过程可以通过另一个元函数 sapply( ) 来实现。函数 sapply( ) 的第一个参数是一个向量，第二个参数为一个函数。结果是另一个向量，该向量和第一个参数有相同的长度，元素为第二个参数中的函数应用到第一个参数中向量的每一个元素后得到的结果。这意味着 sapply( ) 的结果将是填补变量 PO4 缺失值的向量。最后一条赋值语句是使用函数复合的另一个例子。事实上，它等价于先用函数 is. na( ) 的结果对数据框的行进行索引，然后对选择结果的每一个元素通过函数 sapply( ) 应用函数 fillPO4( )。

58

对线性关系的研究使我们能够填充一些新的缺失值。然而，还有几个观测值含有缺失值。可以试着探索案例数据中含有缺失值的变量和名义变量之间的关系。这可以通过应用 R 添加包 lattice 中的函数来绘制条件直方图来进行。如图 2-7 所示，绘图的代码如下：

```
> histogram(~mxPH | season, data = algae)
```

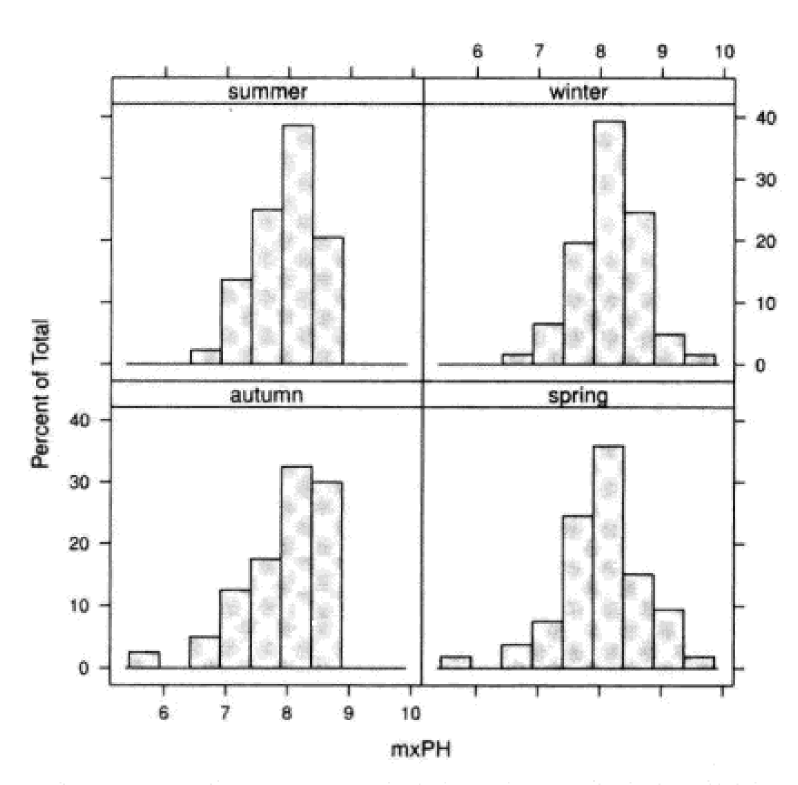


图 2-7　在变量 season 条件下的变量 mxPH 的直方图

上面代码绘制在不同季节变量 mxPH 的直方图。每个直方图对应于某个季节的观测值数据。注意，图 2-7 中的季节顺序不是按照自然的时间顺序，可以转换数据框中因子季节标签的顺序，这样可以使图形中的季节值为自然时间顺序。代码如下：

59

```
> algae$season <- factor(algae$season, levels = c("spring",
+      "summer", "autumn", "winter"))
```

默认情况下，当把名义变量的值转换为因子时，参数 levels 假定因子的水平值按照字母顺序排列。在本案例中，需要因子水平的不同顺序（即季节的时间顺序），所以在因子函数中需要指定因子水平的排序方式。试着执行以上命令，得到新输出的直方图后，看看有什么不同。

注意，图 2-7 中的直方图十分类似，因此收集样本时该年的季节对变量 mxPH 的值没有显著影响。如果对河流的大小（变量 size）进行上面类似的分析，执行指令 histogram（ ~ mxPH |

size, data = algae), 那么从得到的直方图中可知较小的河流有较小的 mxPH 值。对这种相关性的研究可以扩展到多个名义变量，例如：

```
> histogram(~mxPH | size * speed, data = algae)
```

它说明河流大小和速度的所有组合的 mxPH 值的变化。需要指出的是，它没有说明速度较低且规模较小河流的相关信息[⊖]。仅有一个样本具备这些性质，即第 48 条样本，而变量 mxPH 在该样本上的值恰恰缺失！

通过另一种方式也可以获得类似的信息，但这次应用变量的具体取值：

```
> stripplot(size ~ mxPH | speed, data = algae, jitter = T)
```

上面指令的结果如图 2-8 所示。参数 jitter = T 说明对 Y 轴的变量值进行小范围的随机排列，这样可以避免相同值之间的相互重叠而导致失去具有某些特定值的观察值集中度的信息。

这种类型的分析可以应用到其他含有缺失值的变量中。而且，这种分析是一个繁琐的过程，它们有太多的变量组合需要分析。不过，这种方法可以应用到有少量名义变量的较小数据集的分析中。

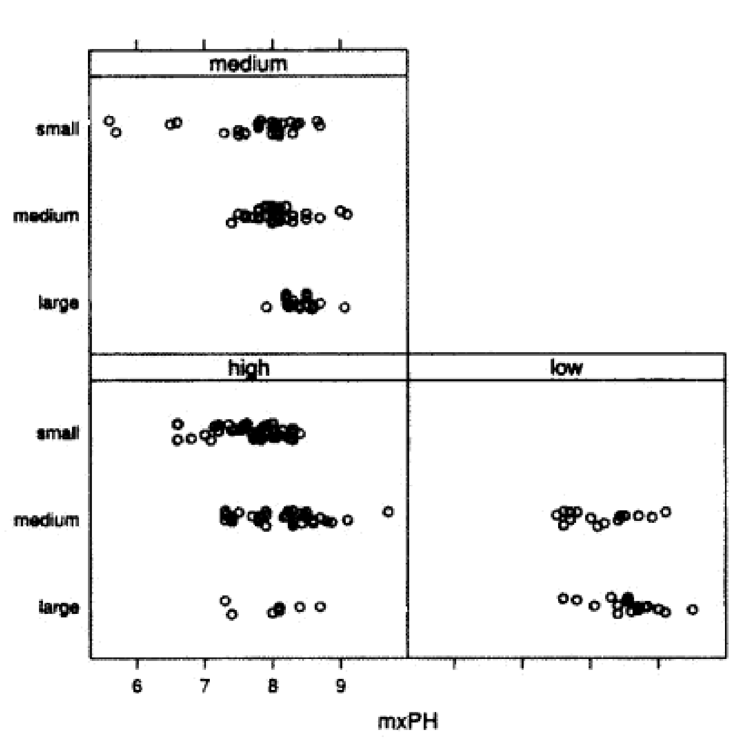


图 2-8　词流大小和速度引起的 mxPH 的变化

2.5.4　通过探索案例之间的相似性来填补缺失值

不同于探索数据集列（变量）之间的相关性，本节尝试使用行（观察值）之间的相似性来填补缺失值。我们可以使用这种方法来填补除去那两个含有太多 NA 值的样本外的其他缺失数据。再次载入本案例数据来覆盖本节之前部分的代码（假设你已经运行了前面的代码）。

60

```
> data(algae)
> algae <- algae[-manyNAs(algae), ]
```

本节所描述的方法假设如果两个水样是相似的，其中一个水样在某些变量上有缺失值，那么该缺失值很可能与另一个水样的值是相似的。为了使用这种直观的方法，首先定义相似性概念。相似性经常由描述观察值的多元度量空间的变量所定义。在文献中有许多度量相似性的指标，常用的是欧式距离。这个距离可以非正式地定义为任何两个案例的观测值之差的平方和。计

⊖ 前面给出了变量 mxPH 的均值来填充它的缺失值的命令。如果执行了该命令，就不是这里的情形了。

算公式如下：

$$d(\boldsymbol{x},\boldsymbol{y}) = \sqrt{\sum_{i=1}^{p}(x_i - y_i)^2} \tag{2-1}$$

下面要描述的方法将使用这种度量来寻找与任何含有缺失值的案例最相似的 10 个水样，并用它们来填补缺失值。我们考虑两种应用这些值的方法。第一种方法简单地计算这 10 个最相近的案例的中位数并用这个中位数来填补缺失值。如果缺失值是名义变量（本案例的 algae 数据不存在这种情况），我们采用这 10 个最相似数据中出现次数最多的值（即众数）。第二种方法采用这些最相似数据的加权均值。权重的大小随着距待填补缺失值的个案的距离增大而减小。这里 61 用高斯核函数从距离获得权重。如果相邻个案距待填补缺失值的个案的距离为 d，则它的值在加权平均中的权重为：

$$w(d) = e^{-d} \tag{2-2}$$

上面的方法可以通过本书添加包中的函数 knnImputation( ) 来实现。这个函数用一个欧式距离的变种来找到距任何个案最近的 k 个邻居。这个变种的欧式距离可以应用于同时含有名义变量和数值变量的数据集中。计算公式如下：

$$d(\boldsymbol{x},\boldsymbol{y}) = \sqrt{\sum_{i=1}^{p}\delta_i(x_i, y_i)} \tag{2-3}$$

其中 $\delta_i()$ 是变量 i 的两个值之间的距离，即

$$\delta_i(v_1, v_2) = \begin{cases} 1 & \text{当 } i \text{ 是名义变量且 } v_1 \neq v_2 \text{ 时} \\ 0 & \text{当 } i \text{ 是名义变量且 } v_1 = v_2 \text{ 时} \\ (v_1 - v_2)^2 & \text{当 } i \text{ 是数值变量时} \end{cases} \tag{2-4}$$

在计算距离时，一般要对数值变量进行标准化，即

$$y_i = \frac{x_i - \bar{x}}{\sigma_x} \tag{2-5}$$

下面说明如何使用 knnImputation( ) 函数。

```
> algae <- knnImputation(algae, k = 10)
```

如果用中位数来填补缺失值，可以使用如下代码

```
> algae <- knnImputation(algae, k = 10, meth = "median")
```

总之，通过这些简单的操作，数据集中不再含有 NA 值（缺失值），为使用 R 的其他函数进行分析做好充分的准备工作。

当决定用前面介绍的哪种方法来填补缺失值时，大多数时候应该根据所分析领域的知识来确定。根据个案之间的相似性来填补缺失值看起来更合理，但这种方法也存在其他问题，例如可能存在不相关的变量扭曲相似性，甚至造成大型数据集的计算特别复杂等问题。另外，对于这些 62 大数据集问题，可以通过随机抽取样本的方法来计算它们之间的相似性。

处理缺失值的参考文献

Pyle（1999）的《Data Preparation for Data Mining》一书有关于数据挖掘的数据准备的所有事项的大量信息，其中就包括处理缺失值的内容。Weiss 和 Indurkhya（1999）的《Predictive Data Mining》一书中有通用的数据准备，特别是缺失数据的很好内容。

Hong（1997）以及 Wilson 和 Martinez（1997）的文章是有关不同类型的变量间距离衡量很好的参考。更进一步的参考文献可以在 Torgo（1999a）的论文中找到。

2.6 获取预测模型

本案例的主要研究目的是预测 140 个水样中 7 种海藻的出现频率。假设海藻频率是数值型数

据，因此可以考虑进行回归分析[一]。简单地说，预测任务是建立一个模型来找到一个数值变量和一组解释变量的关系。这个模型既可以根据未来解释变量的值来预测目标变量，也可以帮助更好地理解问题中各个变量之间的相互联系。

本节将研究适用于预测海藻的两种不同的模型：多元线性回归模型和回归树模型。这里主要基于本书中的问题进行示例性的选择模型，而非基于严格的模型选择步骤。不过，这两种模型是应用于回归问题的较好模型，因为它们对逼近的回归函数的形式有完全不同的假设，两类模型都易于解释，而且可以在任何计算机上非常快速地运行。但这并不意味着在进行数据挖掘时不能尝试别的模型，然后用一些严格的模型选择方法（参见 2.7 节）来选择其中的一个或者多个用于最后预测 140 个测试数据。

这两个模型以不同的方式解决缺失值问题。R 中的线性回归不能使用有缺失值的数据集，而回归树模型可以很自然地处理这些带有缺失值的数据。因此，在建立模型之前，使用不同的方法来进行数据准备工作。对线性回归模型，我们运用 2.5 节中描述的方法进行数据预处理，然后应用线性回归模型。在回归树模型中，我们将直接应用原始的 200 个水样记录[二]。

63

在下面的分析中，假定 140 个测试水样的目标变量的真实值是未知的。因为之前提到，本书网站中包括了预测问题的答案，这些答案给出了我们最终模型的结果以供读者参考。

2.6.1 多元线性回归

多元线性回归模型是最常用的统计数据分析方法，该模型给出了一个有关目标变量与一组解释变量关系的线性函数。这个线性函数是形如 $\beta_i \times X_i$ 这样的项的和，这里 X_i 是预测变量，β_i 是一个常数。

正如之前提到的，在多元线性回归模型中没有处理缺失值的方法。因此，这里将应用根据训练集数据个案的相似性（参见 2.5.4 节）方法来填补缺失值。要注意的是，在使用这种填补方法之前，首先移除第 62 条和第 199 条水样记录，因为在这两条记录的 11 个预测变量中有 6 个是有缺失值的。以下代码获取一个不含有缺失值的数据框：

```
> data(algae)
> algae <- algae[-manyNAs(algae), ]
> clean.algae <- knnImputation(algae, k = 10)
```

在运行上面的代码后，得到的数据框 clean. algae 将不含有缺失值。

接下来，将建立一个用于预测海藻频率的线性回归模型：

```
> lm.a1 <- lm(a1 ~ ., data = clean.algae[, 1:12])
```

函数 lm( ) 建立一个线性回归模型，其中的第一个参数给出了模型的函数形式[三]。在这个例子中，函数的形式是用数据中的其他所有变量来预测变量 $a1$，第一个参数中的点子“.”代表数据框中的所有除 $a1$ 外的变量。如果需要用预测变量 mxPH 和 NH4 来预测变量 $a1$，就要定义模型为“a1 ~ mxPH + NH4”。还有许多其他定义模型的方式，这都称为 R 公式，后边用到时将进行介绍。参数 data 是用来设定建模所用的数据集[四]。

64

函数 lm( ) 的结果是一个含有线性模型信息的对象。可以通过下列代码获取更多线性模型的信息：

[一] 实际上，由于我们要预测每种水样的 7 种频率值，所以我们可以把这个问题分成 7 种不同的线性回归。

[二] 实际上，由于缺失值太多，我们只需要移除两个缺失值。

[三] 其实，R 中用于建立模型的大部分函数都是如此。

[四] 我们已经指出了 11 种解释变量加上表示海藻 $a1$ 列。

```
> summary(lm.a1)

Call:
lm(formula = a1 ~ ., data = clean.algae[, 1:12])

Residuals:
    Min      1Q  Median      3Q     Max
-37.679 -11.893  -2.567   7.410  62.190

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  42.942055  24.010879   1.788  0.07537 .
seasonspring  3.726978   4.137741   0.901  0.36892
seasonsummer  0.747597   4.020711   0.186  0.85270
seasonwinter  3.692955   3.865391   0.955  0.34065
sizemedium    3.263728   3.802051   0.858  0.39179
sizesmall     9.682140   4.179971   2.316  0.02166 *
speedlow      3.922084   4.706315   0.833  0.40573
speedmedium   0.246764   3.241874   0.076  0.93941
mxPH         -3.589118   2.703528  -1.328  0.18598
mnO2          1.052636   0.705018   1.493  0.13715
Cl           -0.040172   0.033661  -1.193  0.23426
NO3          -1.511235   0.551339  -2.741  0.00674 **
NH4           0.001634   0.001003   1.628  0.10516
oPO4         -0.005435   0.039884  -0.136  0.89177
PO4          -0.052241   0.030755  -1.699  0.09109 .
Chla         -0.088022   0.079998  -1.100  0.27265
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 17.65 on 182 degrees of freedom
Multiple R-squared: 0.3731,     Adjusted R-squared: 0.3215
F-statistic: 7.223 on 15 and 182 DF,  p-value: 2.444e-12
```

在解释函数 summary( ) 应用到线性模型对象所得到的信息之前，先介绍 R 如何处理 3 个名义变量。当像上面一样进行模型构建时，R 会生成一组的辅助变量[⊖]，即对每一个有 k 个水平的因子变量，R 会生成 $k-1$ 个辅助变量。这些辅助变量的值为 0 或者 1。当辅助变量的值为 1，表明该因子值出现，同时表明所有其他辅助变量的值为 0。如果所有这 $k-1$ 个辅助变量取值都为 0，则表明因子变量的取值为第 k 个剩余的值。在以上的汇总结果中，可以看到 R 为因子变量 season 生成了 3 个辅助变量（seasonspring、seasonsummer 和 seasonwinter）。如果某个水样的 season 变量的取值为“auumn”，则所有 3 个辅助变量的值将全部为零。

65

对得到的线性模型对象应用函数 summary( )，将给出所建立模型的一些诊断信息。首先是有关线性模型中数据拟合的残差。残差应该是均值为 0 并且为正态分布。（显然残差最好尽可能地小！）

对于每个多元线性回归方程的系数（变量），R 显示它的估计值和标准误差（这些系数变化程度的估计）。为了检验这些系数的重要性，可以进行这些系数为 0 的假设检验，即 $H_0:\beta_i=0$。通常使用 t 检验来验证这些假设。R 计算 t 值，该值定义为估计系数值与其标准误差的比，即$\frac{\beta_i}{s_{\beta_i}}$。R 将显示与系数相关联的一列（$Pr(>|t|)$）表示系数为 0 这一假设被拒绝的概率。因此，该

⊖ 有时也称为虚拟变量。

值为 0.0001 表明有 99.99% 的置信度认为这个系数并非为 0。对于每个测试，R 都给出一个标志来表示相对应的测试置信度水平。总之，仅对于这些在前面有标志的系数，我们至少有 90% 的置信度来拒绝系数为 0 这一假设。

另一个由 R 输出的模型诊断信息是 R^2（或者多元 R^2 或调整 R^2）。R^2 表明模型与数据的吻合度，即模型所能解释的数据变差的比例。R^2 越近于 1（几乎 100% 地解释模型数据的变差）就说明模型拟合得越好；R^2 越小，说明模型拟合得越差。调整系数则更严格，它考虑回归模型中参数的数量。

最后，我们还可以检验任何解释变量与目标变量没有依赖关系这一原假设，即 $H_0:\beta_1=\beta_2=\ldots\beta_m=0$。可以通过把 R 给出的 F 统计值与一个临界值进行比较来进行检验。R 提供一个拒绝原假设的置信度水平。因此 p 值为 0.0001 表示有 99.99% 的置信度确定原假设是错误的。通常，如果一个模型不能通过这个检验（即得到的 p 值被认为太大，例如大于 0.1），则单个系数的 t 检验没有意义。

有些诊断信息也可以通过绘制线性模型来进行检验。实际上，可以用一个类似 plot(lm.a1) 的命令来得到一系列的线性模型图，它们有助于了解模型的性能。其中的一个图形绘制拟合的目标变量值和模型残差的散点图。误差相对较大时，R 通常在该散点图中添加该误差相应的行数，这样就可以方便地检查这些误差较大的记录。R 给出的另外一个图形是误差的正态 Q-Q 图，通过它可以检查误差是否符合应有的正态分布[⊖]。

66

该模型解释的方差比例还不是很理想（大约 32%）。还可以拒绝目标变量不依赖于预测变量的假设（F 检验的 p 值很小）。检查某些系数的显著性，可能会质疑有些变量是否应该进入模型中。有多种方法可以用来精简回归模型。本节将介绍向后消元法。

首先用函数 anova() 来精简线性模型。当将 anova() 应用到简单线性模型时，这个函数提供一个模型拟合的方差序贯分析。也就是说，随着公式中项数的增加，模型的残差平方和减少。对前面建立的模型进行方差分析，结果如下。

```
> anova(lm.a1)

Analysis of Variance Table

Response: a1
           Df Sum Sq Mean Sq F value    Pr(>F)
season      3     85    28.2  0.0905 0.9651944
size        2  11401  5700.7 18.3088  5.69e-08 ***
speed       2   3934  1967.2  6.3179 0.0022244 **
mxPH        1   1329  1328.8  4.2677 0.0402613 *
mnO2        1   2287  2286.8  7.3444 0.0073705 **
Cl          1   4304  4304.3 13.8239 0.0002671 ***
NO3         1   3418  3418.5 10.9789 0.0011118 **
NH4         1    404   403.6  1.2963 0.2563847
oPO4        1   4788  4788.0 15.3774 0.0001246 ***
PO4         1   1406  1405.6  4.5142 0.0349635 *
Chla        1    377   377.0  1.2107 0.2726544
Residuals 182  56668   311.4
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

上面结果表明变量 season 对减少模型拟合误差的贡献最小。下面将它从模型中剔除：

⊖ 在理想情况下，所有的误差将在图上显示为直线。

```
> lm2.a1 <- update(lm.a1, . ~ . - season)
```

函数 update( ) 用于对已有的线性模型进行微小的调整。在上面的代码中，应用函数 update( ) 从模型 lm. a1 中移除变量 season 以得到一个新的模型。新模型的汇总信息如下：

67

```
> summary(lm2.a1)

Call:
lm(formula = a1 ~ size + speed + mxPH + mnO2 + Cl + NO3 + NH4 +
    oPO4 + PO4 + Chla, data = clean.algae[, 1:12])

Residuals:
    Min      1Q  Median      3Q     Max
-36.460 -11.953  -3.044   7.444  63.730

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 44.9532874 23.2378377   1.934  0.05458 .
sizemedium   3.3092102  3.7825221   0.875  0.38278
sizesmall   10.2730961  4.1223163   2.492  0.01358 *
speedlow     3.0546270  4.6108069   0.662  0.50848
speedmedium -0.2976867  3.1818585  -0.094  0.92556
mxPH        -3.2684281  2.6576592  -1.230  0.22033
mnO2         0.8011759  0.6589644   1.216  0.22561
Cl          -0.0381881  0.0333791  -1.144  0.25407
NO3         -1.5334300  0.5476550  -2.800  0.00565 **
NH4          0.0015777  0.0009951   1.586  0.11456
oPO4        -0.0062392  0.0395086  -0.158  0.87469
PO4         -0.0509543  0.0305189  -1.670  0.09669 .
Chla        -0.0841371  0.0794459  -1.059  0.29096
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 17.57 on 185 degrees of freedom
Multiple R-squared: 0.3682,      Adjusted R-squared: 0.3272
F-statistic: 8.984 on 12 and 185 DF,  p-value: 1.762e-13
```

新模型的拟合指标 R^2 提高到了 32.8%，仍然不是太理想。下面使用 anova( ) 函数对两个模型进行比较正式的比较，但这次使用两个模型作为参数：

```
> anova(lm.a1,lm2.a1)

Analysis of Variance Table

Model 1: a1 ~ season + size + speed + mxPH + mnO2 + Cl + NO3 + NH4 +
    oPO4 + PO4 + Chla
Model 2: a1 ~ size + speed + mxPH + mnO2 + Cl + NO3 + NH4 + oPO4 +
    PO4 + Chla
  Res.Df   RSS Df Sum of Sq      F Pr(>F)
1    182 56668
2    185 57116 -3      -448 0.4792 0.6971
```

上面的函数通过 F 检验对两个模型进行方差分析，据此评估两个模型是否有显著不同。这种情况下，尽管误差平方和减少了（-448），但是比较结果说明两者的差距并不显著（显著性值 0.6971 说明两个模型不同的可能性有 30%）。注意，新模型比较简单。为了检查能否移除更多的系数，我们再次对 lm2. a1 模型使用 anova( ) 函数。不断重复这个过程直到没有可剔除的候选系数。为了简化向后消元过程，R 有一个函数来执行上面所有过程。

68

下面的代码对初始模型（lm. a1）用向后消元方法得到一个新的线性模型[⊖]。

```
> final.lm <- step(lm.a1)

Start:  AIC= 1151.85
 a1 ~ season + size + speed + mxPH + mnO2 + Cl + NO3 + NH4 + oPO4 +
    PO4 + Chla

         Df Sum of Sq   RSS   AIC
- season  3       425 57043  1147
- speed   2       270 56887  1149
- oPO4    1         5 56623  1150
- Chla    1       401 57018  1151
- Cl      1       498 57115  1152
- mxPH    1       542 57159  1152
<none>                56617  1152
- mnO2    1       650 57267  1152
- NH4     1       799 57417  1153
- PO4     1       899 57516  1153
- size    2      1871 58488  1154
- NO3     1      2286 58903  1158

Step:  AIC= 1147.33
 a1 ~ size + speed + mxPH + mnO2 + Cl + NO3 + NH4 + oPO4 + PO4 +
    Chla

        Df Sum of Sq   RSS   AIC
- speed  2       213 57256  1144
- oPO4   1         8 57050  1145
- Chla   1       378 57421  1147
- mnO2   1       427 57470  1147
- mxPH   1       457 57500  1147
- Cl     1       464 57506  1147
<none>               57043  1147
- NH4    1       751 57794  1148
- PO4    1       859 57902  1148
- size   2      2184 59227  1151
- NO3    1      2353 59396  1153

...
...

Step:  AIC= 1140.09
 a1 ~ size + mxPH + Cl + NO3 + PO4

       Df Sum of Sq   RSS   AIC
<none>              58432  1140
- mxPH  1       801 59233  1141
- Cl    1       906 59338  1141
- NO3   1      1974 60405  1145
- size  2      2652 61084  1145
- PO4   1      8514 66946  1165
```

函数 step( ) 使用 Akaike 信息标准进行模型搜索。默认情况下，搜索使用向后消元方法，但通过设置参数 direction，可以采用其他的方法（参考该函数的帮助文档以获取更多信息）。

⊖ 因为空间问题，所以在此处省略了部分 step( ) 函数的输出。

可以通过下面的代码来获得最后模型的信息：

```
> summary(final.lm)

Call:
lm(formula = a1 ~ size + mxPH + Cl + NO3 + PO4, data = clean.algae[,
    1:12])

Residuals:
    Min      1Q  Median      3Q     Max
-28.874 -12.732  -3.741   8.424  62.926

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 57.28555   20.96132   2.733  0.00687 **
sizemedium   2.80050    3.40190   0.823  0.41141
sizesmall   10.40636    3.82243   2.722  0.00708 **
mxPH        -3.97076    2.48204  -1.600  0.11130
Cl          -0.05227    0.03165  -1.651  0.10028
NO3         -0.89529    0.35148  -2.547  0.01165 *
PO4         -0.05911    0.01117  -5.291 3.32e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 17.5 on 191 degrees of freedom
Multiple R-squared: 0.3527,        Adjusted R-squared: 0.3324
F-statistic: 17.35 on 6 and 191 DF,  p-value: 5.554e-16
```

这个模型所解释的方差比例（R^2）仍然不是很可观，这样的 R^2 表明对海藻案例应用假定的线性模型是不合适的。

69～70

多元线性回归的参考文献

线性回归是最常用的统计技巧之一。因此，大部分的统计学书籍都有这一主题的内容。对于深入的研究，可以参阅更专业的书籍。其中的两本涉及回归分析深入内容的书籍是 Drapper 和 Smith（1981）、Myers（1990）的书籍，它们涵盖了你需要知道的线性回归的绝大部分内容。

2.6.2 回归树

本节给出 R 中的另一种回归模型。即本节描述了如何通过建立回归树（参见 Breiman et al.，1984）来预测海藻 $a1$ 出现的频率。由于这类模型能够处理缺失值，所以这里只需要如前面所述移除 62 号和 199 号水样即可。

建立回归树模型的代码如下：

```
> library(rpart)
> data(algae)
> algae <- algae[-manyNAs(algae), ]
> rt.a1 <- rpart(a1 ~ ., data = algae[, 1:12])
```

第一条指令用于加载 R 中的 rpart 添加包（Therneau and Atkinson，2010），该包中有回归树的实现[⊖]。最后一条指令用于获取回归树。注意，这里函数应用的参数形式与 lm( ) 函数的参数形式相同。函数 rpart( ) 的第二个参数给出用于建立回归树的数据集。

对象 rt. a1 的内容如下：

```
> rt.a1

n= 198
```

⊖ 实际上，还有另外一个添加包也可以实现此类模型，但是在这个案例中，我们只使用 rpart 程序。

```
node), split, n, deviance, yval
      * denotes terminal node
1) root 198 90401.290 16.996460
  2) PO4>=43.818 147 31279.120  8.979592
    4) Cl>=7.8065 140 21622.830  7.492857
      8) oPO4>=51.118 84  3441.149  3.846429 *
      9) oPO4< 51.118 56 15389.430 12.962500
       18) mnO2>=10.05 24  1248.673  6.716667 *
       19) mnO2< 10.05 32 12502.320 17.646870
         38) NO3>=3.1875 9   257.080  7.866667 *
         39) NO3< 3.1875 23 11047.500 21.473910
           78) mnO2< 8 13  2919.549 13.807690 *
           79) mnO2>=8 10  6370.704 31.440000 *
    5) Cl< 7.8065 7  3157.769 38.714290 *
  3) PO4< 43.818 51 22442.760 40.103920
    6) mxPH< 7.87 28 11452.770 33.450000
     12) mxPH>=7.045 18  5146.169 26.394440 *
     13) mxPH< 7.045 10  3797.645 46.150000 *
    7) mxPH>=7.87 23  8241.110 48.204350
     14) PO4>=15.177 12  3047.517 38.183330 *
     15) PO4< 15.177 11  2673.945 59.136360 *
```

回归树是对某些解释变量分层次的逻辑测试。基于树的模型自动筛选某些相关的变量，这样导致不是所有的变量都会在树中出现。树从 R 标为 1 的根结点开始读，R 在这个结点中提供数据的相关信息。即，可以在该结点中看到一共有 198 个水样（用于构建树的训练集数据样本量），在这 198 个水样中，海藻 *a*1 出现的平均频率为 16.99，相对平均值的偏差[⊖]为 90 401.29。树的每个结点有两个分支，这与预测变量的检验结果有关。例如，在根结点中有一个相应于测试“PO4 ≥ 43.818”为真（含有 147 个水样）的个案分支（R 输出中标为“2”），同时也有另一个分支包含剩余的 51 个不满足这个测试的水样（R 标记为“3”）。从结点 2 有两个分支分别连接到结点 4 和结点 5，具体到哪个结点由对变量 Cl 的检验来确定。不断进行以上的检验，直到达到某一个叶结点，这些叶结点在 R 中由星号标记出来。在叶结点，我们就可以对树进行预测了。也就是说，如果我们想建立一个回归树来预测某个水样的频率，只要从根结点开始根据对该水样检验的结果，追踪某个分支，直到叶结点。叶结点目标变量的平均值就是树的预测值。

我们也可以得到回归树的图形表示。可以用函数 plot() 和函数 text() 对树对象绘图即可。这两个函数有多个参数来控制树的可视化。为了方便地得到漂亮的树的可视化图形，本书的 R 添加包中提供了函数 prettyTree()。对上面得到的树对象应用该函数，得到图形如图 2-9 所示。

```
> prettyTree(rt.a1)
```

函数 summary() 也可以用于树对象。此函数将给出许多有关于树的测试信息、其他可能考虑的测试以及中间分割等。这里的中间分割是 R 回归树处理缺失值的一种方法。

通常分为两步来建立回归树。最初，生成一棵较大的树，然后通过统计估计删除底部的一些结点来对树进行修剪。这个过程的目的是防止过度拟合。事实上，一个过度大的树一般会很好地对训练集数据进行拟合，但是它会拟合给定数据集中的一些虚假的关系，因此当把该模型用于新数据的预测时，预测性能很差。在许多建模技术中存在过度拟合问题，尤其是当需要逼近的函数的假设条件不是很严格的时候。对于要求不严格的模型，虽然它们的要求不高，有广泛的应用范围，但却存在过度拟合问题，所以它需要一个事后统计估计步骤来避免过度拟合问题。

⊖ 不同值与平均值之差的平方和。

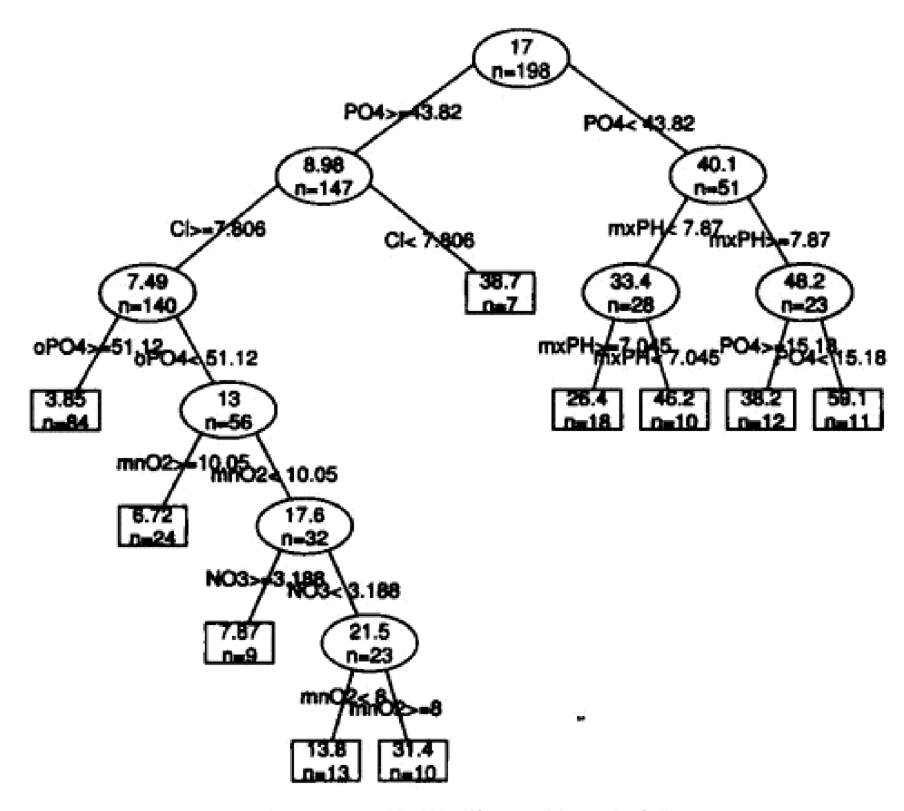


图 2-9　预测海藻 $a1$ 的回归树

上面使用 rpart( ) 函数构建树，在构建树的过程中，当给定条件满足时构建过程就停止。当下列条件满足时，树构建过程将结束：1）偏差的减少小于某一个给定界限值时；2）当结点中的样本数量小于某个给定界限时；3）当树的深度大于一个给定的界限值。上面 3 个界限值分别由 rpart( ) 函数的三个参数（cp、minsplit、maxdepth）来确定。它们的默认值分别为 0.01、20 和 30。如果要避免树的过度拟合问题，就要经常检查这些默认值的有效性。这可以通过对得到的树采取事后修剪过程来进行。

rpart 添加包实现了一种称为*复杂度损失修剪*的修剪方法（Breiman et al.，1984）。这个方法使用 R 在每个树结点计算的参数值 cp。这种修剪方法试图估计 cp 值以确保达到预测的准确性和树的大小之间的最佳折中。给出一个由函数 rpart( ) 建立的回归树，R 可以生成这棵树的一些子树，并估计这些树的性能。这些信息可以通过函数 printcp( ) 得到[⊖]：

```
> printcp(rt.a1)

Regression tree:
rpart(formula = a1 ~ ., data = algae[, 1:12])

Variables actually used in tree construction:
[1] Cl   mnO2 mxPH NO3  oPO4 PO4

Root node error: 90401/198 = 456.57

n= 198

        CP nsplit rel error  xerror    xstd
1 0.405740      0   1.00000 1.00932 0.12986
2 0.071885      1   0.59426 0.73358 0.11884
3 0.030887      2   0.52237 0.71855 0.11518
4 0.030408      3   0.49149 0.70161 0.11585
```

⊖　可以通过函数 plotcp(rt，a1) 以图形方式来得到类似的信息。

```
5 0.027872        4    0.46108 0.70635 0.11403
6 0.027754        5    0.43321 0.69618 0.11438
7 0.018124        6    0.40545 0.69270 0.11389
8 0.016344        7    0.38733 0.67733 0.10892
9 0.010000        9    0.35464 0.70241 0.11523
```

由 rpart() 函数建立的回归树是上面列表中的最后一个树（树 9）。这个树的 cp 值为 0.01（参数 cp 的默认值），该树包括九个测试和一个相对误差值（与根结点相比）0.354。然而，R 应用 10 折交叉验证的内部过程，评估该树的平均相对误差[⊖]为 0.702 41 ±0.115 23。根据这些更稳健的性能估计信息，可以避免过度拟合问题。可以看到，8 号树的预测相对误差（0.677 33）最小。另一个选择标准是根据 1-SE 规则来选择最好的回归树，这包括检查交叉验证的估计误差（“xerror”列），以及标准误差（“xstd”列）。在这个案例中，1-SE 规则树是最小的树，误差小于 0.677 33 +0.108 92 =0.786 25，而由 1 检验的 2 号树的估计误差为 0.733 58。如果我们选择这个树而不是 R 建议的树，我们就可以通过使用不同的 cp 值[⊜]来建立这棵树。

```
> rt2.a1 <- prune(rt.a1, cp = 0.08)
> rt2.a1

n= 198

node), split, n, deviance, yval
      * denotes terminal node

1) root 198 90401.29 16.996460
  2) PO4>=43.818 147 31279.12  8.979592 *
  3) PO4< 43.818 51 22442.76 40.103920 *
```

在本书添加包中的 rpartXse() 函数可以自动运行这个过程，它的参数 se 的默认值为 1。

```
> (rt.a1 <- rpartXse(a1 ~ ., data = algae[, 1:12]))

n= 198

node), split, n, deviance, yval
      * denotes terminal node

1) root 198 90401.29 16.996460
2) PO4>=43.818 147 31279.12  8.979592 *
3) PO4< 43.818 51 22442.76 40.103920 *
```

71 ~ 75

可以应用 R 的函数 snip. rpart() 来交互地对树进行修剪。这个函数可以通过两种方式生成一个修剪过的回归树。第一种方法是指出需要修剪那个地方的结点号（可以通过输出树对象来得到树的结点号）：

```
> first.tree <- rpart(a1 ~ ., data = algae[, 1:12])
> snip.rpart(first.tree, c(4, 7))

n= 198

node), split, n, deviance, yval
      * denotes terminal node
```

⊖ 注意，你可能在列“xerror”和列“xstd”得到不同的数值。交互验证估计值是通过随机抽样得到的，这意味着你的抽样可能和这里的不同，因此得到的结果也是不同的。

⊜ 事实上，可以用它对应的 cp 值和它上面的那棵树的 cp 值之间的任何数值。

```
1) root 198 90401.290 16.996460
  2) PO4>=43.818 147 31279.120  8.979592
    4) Cl>=7.8065 140 21622.830  7.492857 *
    5) Cl< 7.8065 7  3157.769 38.714290 *
  3) PO4< 43.818 51 22442.760 40.103920
    6) mxPH< 7.87 28 11452.770 33.450000
     12) mxPH>=7.045 18  5146.169 26.394440 *
     13) mxPH< 7.045 10  3797.645 46.150000 *
    7) mxPH>=7.87 23  8241.110 48.204350 *
```

这个函数与 rpart( ) 函数一样返回一个树对象，所以可以用形如 my. tree <- snip. rpart (first. tree, c(4,7)) 这样的代码来保存这个修剪过的树。

另外，也可以在图形窗口下使用 snip. rpart( ) 函数。首先，画出回归树，然后调用没有第二个参数的函数。如果点击回归树的某些结点，R 会在控制台输出这些结点的信息。如果继续点击这个结点，R 就在这个结点对树进行修剪[⊖]。可以在图形窗口继续修剪回归树，直到右击结束这一交互式的修剪过程。调用该函数的结果仍然是一个树对象：

```
> prettyTree(first.tree)
> snip.rpart(first.tree)

node number: 2   n= 147
    response= 8.979592
    Error (dev) =  31279.12
node number: 6   n= 28
    response= 33.45
    Error (dev) =  11452.77
n= 198

node), split, n, deviance, yval
      * denotes terminal node

1) root 198 90401.290 16.996460
  2) PO4>=43.818 147 31279.120  8.979592 *
  3) PO4< 43.818 51 22442.760 40.103920
    6) mxPH< 7.87 28 11452.770 33.450000 *
    7) mxPH>=7.87 23  8241.110 48.204350
     14) PO4>=15.177 12  3047.517 38.183330 *
     15) PO4< 15.177 11  2673.945 59.136360 *
```

在上例中，点击并修剪了结点 2 和结点 6。

回归树的参考文献

如果需要更加全面的学习回归树，可以参考 Breiman 等(1984) 的书籍。该书是分类树和回归树的标准参考文献。对一些读者而言，本书的方法可能有些太正式（至少某些章节)。无论如何，这本书都绝对是一本极好的参考书，尽管它更偏重于统计文献。从机器学习方面而言，Quinlan (1993) 的有关 C4.5 的书是一本有关分类树的很好的参考书。本书作者的博士论文（Torgo, 1999a）给出了很好的回归树入门知识和高级主题，你可以从作者网站免费下载。论文中也介绍了其他基于树的模型，它们的目的是在叶结点用更复杂的模型来提高回归树的精确度（Torgo, 2000)。

2.7 模型的评价和选择

2.6 节给出了本案例的两个预测模型的例子。最明显的问题是，应该使用哪一个模型来获得

⊖ 注意，因为回归树的图片没有更新，所以你不会在图形窗口看到修剪回归树的过程。

7 种海藻的 140 个测试样品的预测。为了回答这个问题，需要在可供选择的模型空间中指定一些模型的偏好标准，也就是说，需要详细说明应该如何评价模型的性能。

有多种评价（和比较）模型的标准。其中最流行的标准是计算模型的预测性能。当然还有其他衡量模型的标准，例如模型的可解释性，还有对大型数据挖掘特别重要的标准，即模型的计算效率。

回归模型的预测性能是通过将目标变量的预测值与实际值进行比较得到的，并从这些比较中计算某些平均误差的度量。一种度量方法是平均绝对误差（MAE）。下面描述如何获得 2.6 节中两个模型（线性回归和回归树）的平均绝对误差。第一步，获取需要评价模型预测性能的测试集个案的预测值。在 R 中，要获得任何模型的预测，就要使用函数 predict() 进行预测。函数 predict() 是一个泛型函数，它的一个参数为需要应用的模型，另一个参数为数据的测试集，输出结果为相应的模型预测值：

76 ~ 77

```
> lm.predictions.a1 <- predict(final.lm, clean.algae)
> rt.predictions.a1 <- predict(rt.a1, algae)
```

上面两个命令将输出 2.6 节中得到的预测海藻 $a1$ 的两个模型的预测值。注意，因为原始训练集数据含有缺失值，所以在线性回归模型中使用的数据是数据框 clean. algae。

得到模型的预测值后，就可以计算出其平均绝对误差，如下所示：

```
> (mae.a1.lm <- mean(abs(lm.predictions.a1 - algae[, "a1"])))

[1] 13.10681

> (mae.a1.rt <- mean(abs(rt.predictions.a1 - algae[, "a1"])))

[1] 11.61717
```

另一种流行的误差度量是均方误差（MSE）。可以由下列代码计算均方误差：

```
> (mse.a1.lm <- mean((lm.predictions.a1 - algae[, "a1"])^2))

[1] 295.5407

> (mse.a1.rt <- mean((rt.predictions.a1 - algae[, "a1"])^2))

[1] 271.3226
```

后一种误差度量方法的不足之处是：误差值和目标变量的单位不统一，因此从用户的角度看，这种误差不好解释。即使应用平均绝对误差（MAE）来度量误差，问题是如何判断模型的得分是好还是差。能够解决这一问题的误差度量是标准化后的平均绝对误差（NMSE）。这一统计量是计算模型预测性能和基准模型的预测性能之间的比率。通常采用目标变量的平均值来作为基准模型，代码如下：

```
> (nmse.a1.lm <- mean((lm.predictions.a1-algae[,'a1'])^2)/
+                   mean((mean(algae[,'a1'])-algae[,'a1'])^2))

[1] 0.6473034

> (nmse.a1.rt <- mean((rt.predictions.a1-algae[,'a1'])^2)/
+                   mean((mean(algae[,'a1'])-algae[,'a1'])^2))
```

78

```
[1] 0.5942601
```

NMSE 是一个比值，其取值范围通常为 0 ~ 1。如果模型表现优于这个非常简单的基准模型预测，那么 NMSE 应明显小于 1。NMSE 的值越小，模型的性能就越好。NMSE 的值大于 1，意味着

模型预测还不如简单地把所有个案的平均值作为预测值！

在本书提供的 R 添加包中，函数 regr. eval( ) 用来计算线性回归模型的性能度量指标。下面给出应用该函数一个例子。可以查找该函数的帮助文档来获取这个函数的不同用法。

```
> regr.eval(algae[, "a1"], rt.predictions.a1, train.y = algae[,
+     "a1"])

        mae         mse        rmse        nmse        nmae
 11.6171709 271.3226161  16.4718735   0.5942601   0.6953711
```

可视化地查看模型的预测值将更加有趣。一种方法是绘制误差的散点图。图 2-10 给出了对两种模型的预测值的可视化分析，这是由以下的代码产生的：

```
> old.par <- par(mfrow = c(1, 2))
> plot(lm.predictions.a1, algae[, "a1"], main = "Linear Model",
+     xlab = "Predictions", ylab = "True Values")
> abline(0, 1, lty = 2)
> plot(rt.predictions.a1, algae[, "a1"], main = "Regression Tree",
+     xlab = "Predictions", ylab = "True Values")
> abline(0, 1, lty = 2)
> par(old.par)
```

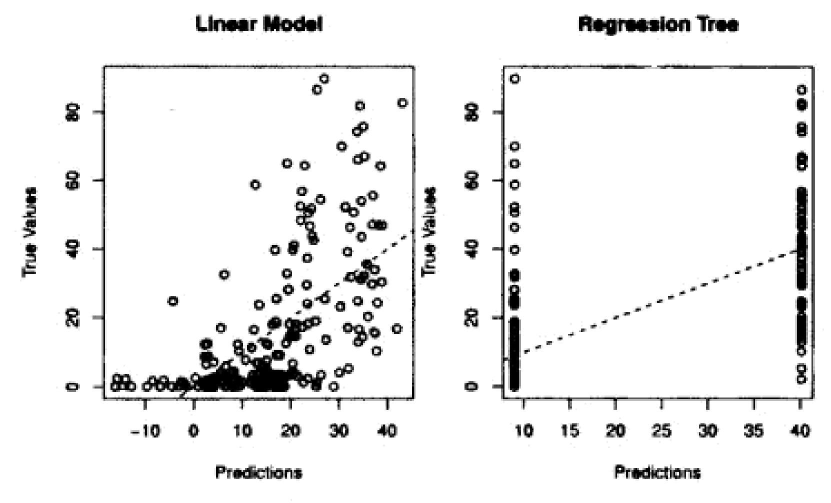


图 2-10 模型预测值和真实值的散点图

从图 2-10 中可知，这两个模型在许多个案上的性能比较差。在理想的情况下，模型对所有的案例做出正确的预测时，图中的所有圈应该在虚线上，这条虚线是通过函数 abline(0,1,lty = 2)来绘制的。这条虚线穿过坐标系的原点，代表 x 坐标与 y 坐标相等的点集。图 2-10 中每个圆圈的 x 坐标和 y 坐标分别代表目标变量的预测值和真实值，如果它们相等，那么这些圆圈就会落在这条理想的直线上。正如从图 2-10 中所观察到的，情况并非如此！可以用函数 identify( ) 来检查那些预测特别差的样本点，该函数可以让用户通过互动方式点击图形中的点，代码如下：

79

```
> plot(lm.predictions.a1,algae[,'a1'],main="Linear Model",
+      xlab="Predictions",ylab="True Values")
> abline(0,1,lty=2)
> algae[identify(lm.predictions.a1,algae[,'a1']),]
```

运行上面的代码，并在图形上点击，然后右击结束交互过程后，应该看到相应于所点击的圆圈的海藻数据框的行数据——因为这里用函数 identify( ) 得到的向量来索引海藻数据框。

观察图 2-10 的左图，它对应的是线性回归模型。注意，有一些个案的海藻频率的预测值为负值。在本案例中，海藻在出现频率为负值时没有意义（至少是零）。因此，可以用以上知识和海藻频率的最小可能取值来优化上面的线性回归模型。

```
> sensible.lm.predictions.a1 <- ifelse(lm.predictions.a1 <
+     0, 0, lm.predictions.a1)
> regr.eval(algae[, "a1"], lm.predictions.a1, stats = c("mae",
+     "mse"))

      mae       mse
 13.10681 295.54069

> regr.eval(algae[, "a1"], sensible.lm.predictions.a1, stats = c("mae",
+     "mse"))

      mae       mse
 12.48276 286.28541
```

上面代码应用函数 ifelse() 来改进模型的预测结果。该函数有三个参数，第一个参数是逻辑条件，第二个参数是当逻辑条件为真时函数的取值，第三个参数是当逻辑条件不成立时函数的取值。注意，通过这一小的细节就提高了模型的性能。

80

根据以上计算出的模型的性能指标，我们倾向于选择回归树模型来预测 140 个测试样品的频率值，因为该模型有较低的 NMSE 值。然而，这种推理有一个缺陷。我们的分析目标是获得能够对 140 个测试样品的频率进行预测的最佳模型。由于不知道这些测试样本的目标变量值，所以我们需要估计哪一个模型将在这些测试样本上有较好的性能。这里的关键问题是在不知道数据集真实的目标变量取值时，要获得模型在该数据集上可靠的性能估计。使用已有的训练数据获得模型的性能指标（如前文所进行的过程）是不可靠的，因为这些计算是有偏的。实际上，有的模型可以很容易地获得训练数据的零误差预测。然而，模型的这一优秀性能很难推广到目标变量值未知的新样本上。正如之前所述，这种现象通常称为过度拟合训练数据。因此，为了选择一个合适的模型，我们需要获得模型在未知数据上预测性能的更加可靠的估计。k 折交叉验证是获得模型性能可靠估计的一种常用方法，它适用于像本案例这样的小数据集。这种方法可以简要介绍如下。首先获取 k 个同样大小的随机训练数据子集。对于这 k 个子集的每一个子集，用除去它之外的其余 $k-1$ 个子集建立模型，然后用第 k 子集来评估这个模型，最后存储模型的性能指标。对其余的每个子集重复以上过程，最后有 k 个性能指标的测量值，这些性能指标是通过在没有用于建模的数据上计算得到的，这也是关键之处。k 折交叉验证估计是这 k 个性能指标的平均。常见的选择是 $k=10$。有时我们会重复进行多次 k 折交叉验证以获得更加可靠的估计。

总之，当面对一项预测任务时，需要做出下列决策：

- 为预测任务选择模型（同一算法的不同参数设定也可以认为是不同的模型）。
- 选择比较模型性能的评估指标。
- 选择获取评估指标的可靠估计的实验方法。

在书提供的 R 添加包中，提供了函数 experimentalComparison()，它用来进行模型的选择和比较任务。它可以和不同的估计方法一起使用，如交叉验证法。这个函数有三个参数：1）用于比较的数据集；2）需要比较的可选模型；3）实验过程中的系数。我们以海藻数据集为例，用它来比较线性回归模型和几个不同的回归树模型。

函数 experimentalComparison() 适用于任何模型和任何数据，在这个意义上，它是一个泛型函数。使用者提供一组实现待比较的模型的函数。其中每一个函数应该对训练集和测试集实现一个完整的“训练 + 测试 + 评估”周期。在评估过程的每一次迭代中，调用这些函数。这些函数应该返回一个向量，其元素为交叉验证中用户需要的性能评估指标值。下面给出两个目标模型的函数：

81

```
> cv.rpart <- function(form,train,test,...) {
+   m <- rpartXse(form,train,...)
+   p <- predict(m,test)
+   mse <- mean((p-resp(form,test))^2)
+   c(nmse=mse/mean((mean(resp(form,train))-resp(form,test))^2))
+ }
> cv.lm <- function(form,train,test,...) {
+   m <- lm(form,train,...)
+   p <- predict(m,test)
+   p <- ifelse(p < 0,0,p)
+   mse <- mean((p-resp(form,test))^2)
+   c(nmse=mse/mean((mean(resp(form,train))-resp(form,test))^2))
+ }
```

在这个示例中，假设用 NMSE 作为线性回归模型和回归树模型的性能评估指标。所有这些用户定义的函数的前三个参数应该是公式、训练数据和测试数据。实验过程调用函数时可以应用的其他参数包括要评估模型所需要的参数。虽然要评估的两个模型应用了完全不同的学习算法，但是两个模型函数都有同样的“训练 + 测试 + 评估”周期。函数的定义中还包括一个特殊参数“...”。这个特殊参数可以用在任意的 R 函数中，它允许一个特定函数具有可变的参数。其实，“...”这个参数结构是一列表，它用来获取传递给函数的前三个命名参数之后的所有参数。这个结构用于给实际模型传递所需要的额外参数（例如在函数 rpartXse( ) 中和函数 lm( ) 中）。这些函数的另一个特殊之处是应用本书添加包提供的函数 resp( )，它用于根据公式获得数据集的目标变量值。

在定义好用于模型学习和测试的函数后，就可以按下列代码进行模型的交叉验证比较：

```
> res <- experimentalComparison(
+             c(dataset(a1 ~ .,clean.algae[,1:12],'a1')),
+             c(variants('cv.lm'),
+               variants('cv.rpart',se=c(0,0.5,1))),
+             cvSettings(3,10,1234))

#####  CROSS VALIDATION  EXPERIMENTAL COMPARISON #####

** DATASET :: a1

++ LEARNER :: cv.lm  variant ->  cv.lm.defaults
Repetition  1
Fold:  1  2  3  4  5  6  7  8  9  10
Repetition  2
Fold:  1  2  3  4  5  6  7  8  9  10
Repetition  3
Fold:  1  2  3  4  5  6  7  8  9  10


++ LEARNER :: cv.rpart  variant ->  cv.rpart.v1
Repetition  1
Fold:  1  2  3  4  5  6  7  8  9  10
Repetition  2
Fold:  1  2  3  4  5  6  7  8  9  10
Repetition  3
Fold:  1  2  3  4  5  6  7  8  9  10


++ LEARNER :: cv.rpart  variant ->  cv.rpart.v2
```

```
Repetition  1
Fold:  1  2  3  4  5  6  7  8  9  10
Repetition  2
Fold:  1  2  3  4  5  6  7  8  9  10
Repetition  3
Fold:  1  2  3  4  5  6  7  8  9  10


++ LEARNER :: cv.rpart  variant ->  cv.rpart.v3
Repetition  1
Fold:  1  2  3  4  5  6  7  8  9  10
Repetition  2
Fold:  1  2  3  4  5  6  7  8  9  10
Repetition  3
Fold:  1  2  3  4  5  6  7  8  9  10
```

像先前提到那样，第一个参数是含有在实验比较中所应用数据集的一个向量。每个数据集的声明形式为 datatset(<formula>,<data frame>,<label>)。函数 experimentalComparison() 的第二个参数包含要研究的可选的模型方法。每一个模型方法通过函数 variant() 来指定，该函数的第一个参数是用户定义的用于“学习 + 测试 + 评估”周期的函数名称。其余的可选参数用来给出估计方法的其他参数的可选值。函数 variantes() 根据所有参数值的组合生成一组可选模型。在上面例子的代码中，模型“cv. lm”采用了默认参数值，而模型“cv. rpart”的参数 se 则给出了不同的取值。这意味着实验将包含回归树的三个版本，这点可以在上面的函数输出中得到确认。函数 experimentalComparison() 的第三个参数是设定交叉验证实验的参数，即 k 折交叉验证过程重复的次数（这里设为 3）、k 的取值（10）、随机数生成器的种子。最后的参数（随机数种子）设定可以保证在必要的情况下可以重现我们的实验（例如更换了训练模型系统）。

82 ~ 83

这个代码调用的结果是一个复杂的对象，它包含实验比较的所有信息。在本书的 R 添加包中提供了多种获取这些信息的函数。例如，下面代码提供了比较结果的概要：

```
> summary(res)

== Summary of a  Cross Validation  Experiment ==

 3 x 10 - Fold Cross Validation run with seed =  1234

* Datasets :: a1
* Learners  ::  cv.lm.defaults, cv.rpart.v1, cv.rpart.v2, cv.rpart.v3

* Summary of Experiment Results:


-> Datataset:  a1

        *Learner: cv.lm.defaults
             nmse
avg     0.7196105
std     0.1833064
min     0.4678248
max     1.2218455
invalid 0.0000000

        *Learner: cv.rpart.v1
             nmse
avg     0.6440843
```

```
std     0.2521952
min     0.2146359
max     1.1712674
invalid 0.0000000

        *Learner: cv.rpart.v2
                nmse
avg     0.6873747
std     0.2669942
min     0.2146359
max     1.3356744
invalid 0.0000000

        *Learner: cv.rpart.v3
                nmse
avg     0.7167122
std     0.2579089
min     0.3476446
max     1.3356744
invalid 0.0000000
```

从结果中可知，其中的一个回归树有最优的平均 NMSE 值。这个 NMSE 值是否明显优于其他模型，目前还不明显，本节的后面将回到这个问题。可以得到这些结果的可视化图形（见图 2-11）。代码如下：

```
> plot(res)
```

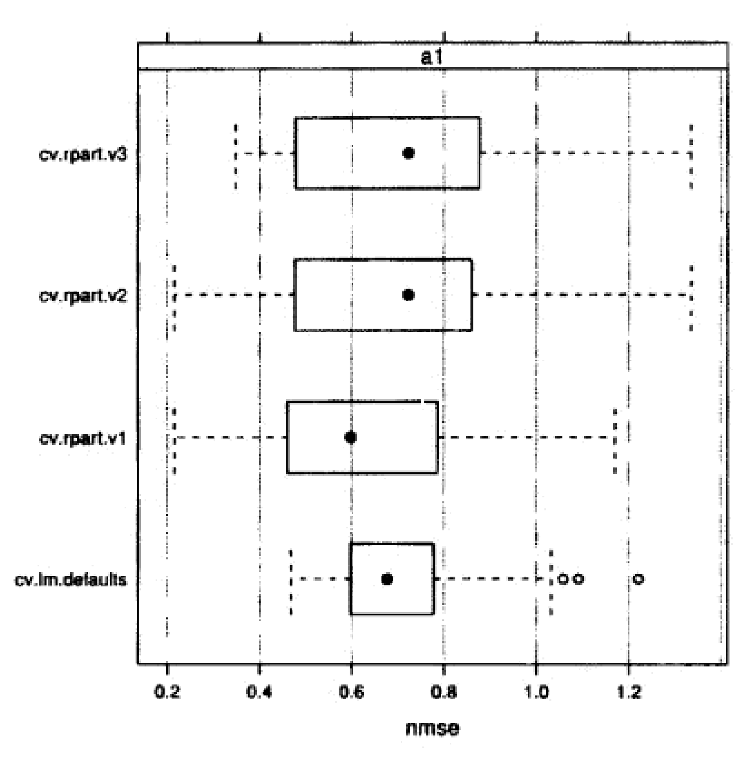


图 2-11　交互验证结果的可视化

函数 experimentalComparison( ) 给每个模型一个标记，如果你想知道任何标记模型所对应的参数，代码如下：

```
> getVariant("cv.rpart.v1", res)
Learner::  "cv.rpart"
Parameter values
         se  =  0
```

可以同时对所有 7 个预测任务进行与上面相似的比较实验。执行以下代码：

```
> DSs <- sapply(names(clean.algae)[12:18],
+          function(x,names.attrs) {
+            f <- as.formula(paste(x,"~ ."))
+            dataset(f,clean.algae[,c(names.attrs,x)],x)
+          },
+          names(clean.algae)[1:11])
> res.all <- experimentalComparison(
+                   DSs,
+                   c(variants('cv.lm'),
+                     variants('cv.rpart',se=c(0,0.5,1))
+                    ),
+                   cvSettings(5,10,1234))
```

为了节省篇幅，我们省略了以上代码的输出。上面代码首先创建用于比较 7 个预测任务的数据集向量。对每一个预测问题需要构建一个公式，该公式由一个字符串构成，它是数据集中相应的需要预测的目标变量和符号“~.”连接而成的。然后，该字符串通过函数 as. formula( ) 转换为一个 R 公式。和前面一样，创建用于函数 experimentalComparison( ) 的数据向量，不同的是要重复 5 次 10 折交叉验证以提高统计结果的显著性。根据计算机的运行速度，这条指令可能要运行一会儿。

图 2-12 展现了在交叉验证方法下，模型对不同海藻的结果。绘制图 2-12 的代码如下：

```
> plot(res.all)
```

从图 2-12 中可知，有几个很差的结果，也就是说，几个 NMSE 值明显大于 1。测试结果比简单地采用目标变量的平均值这一基准模型还要差！如果需要知道每个问题对应的最优模型，可以应用函数 bestScores( )，代码如下：

```
> bestScores(res.all)

$a1
          system   score
nmse cv.rpart.v1 0.64231

$a2
          system score
nmse cv.rpart.v3     1

$a3
          system score
nmse cv.rpart.v2     1

$a4
          system score
nmse cv.rpart.v2     1

$a5
             system     score
nmse cv.lm.defaults 0.9316803

$a6
             system     score
nmse cv.lm.defaults 0.9359697

$a7
              system    score
    nmse cv.rpart.v3 1.029505
```

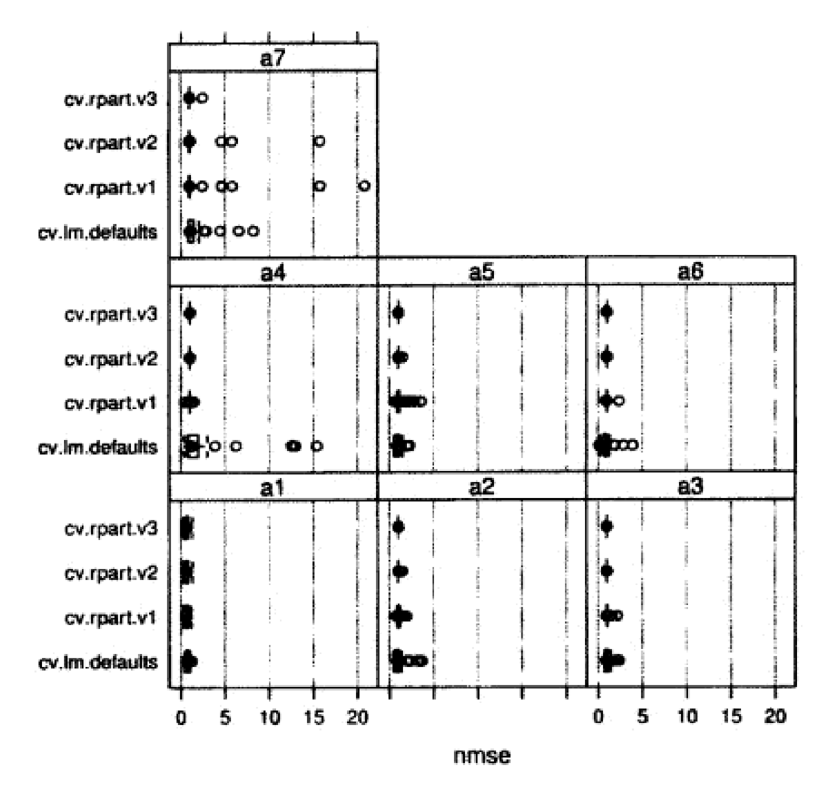


图 2-12 所有海藻的交叉验证结果的可视化

上面结果说明，除了海藻 1 外，其他海藻的预测结果都不好。图 2-12 给出的结果变差表明，组合方法也许是一种好的预测模型。组合法是一种模型构建方法，它通过产生大量可选模型并把这些模型进行组合，这样得到的模型可以克服单个模型的局限性。有许多方法来得到组合模型，这些不同的模型不仅在于获取模型的方法不同（例如，获取模型的训练集数据的不同、变量不同、建模方法不同），也在于组合预测的不同。随机森林（Breiman，2001）被视为组合模型有竞争性的例子之一，它由大量的树模型（回归树或者分类树）构成。每个树是完全生长（没有事后剪枝），在树生长的每一步骤，最好的结点分割方法将从变量集合的一个随机子集中选取。回归任务的预测采用组合中预测结果的平均值。R 的添加包 randomForest（Liaw and Wiener，2002）的函数 randomForest( ) 实现回归树的思想。以下代码是重复先前交叉验证，这次是包含三个版本的随机森林模型，在组合中每个模型有不同数目的树，这里又一次把输出忽略。

```
> library(randomForest)
> cv.rf <- function(form,train,test,...) {
+   m <- randomForest(form,train,...)
+   p <- predict(m,test)
+   mse <- mean((p-resp(form,test))^2)
+   c(nmse=mse/mean((mean(resp(form,train))-resp(form,test))^2))
+ }
> res.all <- experimentalComparison(
+                    DSs,
+                    c(variants('cv.lm'),
+                      variants('cv.rpart',se=c(0,0.5,1)),

+                      variants('cv.rf',ntree=c(200,500,700))
+                     ),
+                    cvSettings(5,10,1234))
```

应用函数 bestScores( )，能证实组合方法的优势：

```
> bestScores(res.all)

$a1
        system     score
nmse cv.rf.v3 0.5447361

$a2
        system     score
nmse cv.rf.v3 0.7777851

$a3
        system     score
nmse cv.rf.v2 0.9946093

$a4
        system     score
nmse cv.rf.v3 0.9591182

$a5
        system     score
nmse cv.rf.v1 0.7907947

$a6
        system     score
nmse cv.rf.v3 0.9126477

$a7
           system    score
nmse cv.rpart.v3 1.029505
```

事实上，除了海藻 7 以外的所有问题，最好的结果是由随机森林的某些变体给出的。而且，结果不是总是很好，尤其是对于海藻 7。函数 bestScores( ) 并没有告诉我们这些最佳模型和剩余其他模型之间的区别是否显著。也就是说，采用另外的随机数据我们能得到相似结果的可能性是多少？本书提供的 R 添加包中的函数 compAnalysis( ) 可以提供这一信息。它对一个模型和其他另一个模型进行成对的 Wilcoxon 检验。下面举例说明该函数的某些应用。

对于海藻 1、2、4 和 6，模型 "cv. rf. v3" 是最好的。下面的代码将给出这一论断的统计显著性：

```
> compAnalysis(res.all,against='cv.rf.v3',
               datasets=c('a1','a2','a4','a6'))


== Statistical Significance Analysis of Comparison Results ==

Baseline Learner::          cv.rf.v3  (Learn.1)

** Evaluation Metric::          nmse

- Dataset: a1
      Learn.1    Learn.2 sig.2    Learn.3 sig.3    Learn.4 sig.4

AVG 0.5447361 0.7077282      ++ 0.6423100      +  0.6569726      ++
STD 0.1736676 0.1639373         0.2399321         0.2397636
      Learn.5 sig.5  Learn.6 sig.6   Learn.7 sig.7
```

```
AVG 0.6875212    ++  0.5490511        0.5454724
STD 0.2348946        0.1746944        0.1766636

- Dataset: a2
      Learn.1   Learn.2 sig.2   Learn.3 sig.3    Learn.4 sig.4
AVG 0.7777851 1.0449317     ++ 1.0426327     ++ 1.01626123     ++
STD 0.1443868 0.6276144        0.2005522        0.07435826
         Learn.5 sig.5   Learn.6 sig.6   Learn.7 sig.7
AVG 1.000000e+00     ++ 0.7829394       0.7797307
STD 2.389599e-16        0.1433550       0.1476815

- Dataset: a4
      Learn.1  Learn.2 sig.2   Learn.3 sig.3      Learn.4 sig.4
AVG 0.9591182 2.111976        1.0073953     +  1.000000e+00     +
STD 0.3566023 3.118196        0.1065607        2.774424e-16
         Learn.5 sig.5   Learn.6 sig.6   Learn.7 sig.7
AVG 1.000000e+00     +  0.9833399       0.9765730
STD 2.774424e-16        0.3824403       0.3804456

- Dataset: a6
      Learn.1   Learn.2 sig.2   Learn.3 sig.3      Learn.4 sig.4
AVG 0.9126477 0.9359697     ++ 1.0191041        1.000000e+00
STD 0.3466902 0.6045963        0.1991436        2.451947e-16
         Learn.5 sig.5   Learn.6 sig.6   Learn.7 sig.7
AVG 1.000000e+00        0.9253011       0.9200022
STD 2.451947e-16        0.3615926       0.3509093

Legends:
Learners -> Learn.1 = cv.rf.v3 ; Learn.2 = cv.lm.defaults ;
Learn.3 = cv.rpart.v1 ; Learn.4 = cv.rpart.v2 ; Learn.5 = cv.rpart.v3 ;
Learn.6 = cv.rf.v1 ; Learn.7 = cv.rf.v2 ;
Signif. Codes -> 0 '++' or '--' 0.001 '+' or '-' 0.05 ' ' 1
```

上面结果中的“sig. X”列提供了我们需要的信息。如果这一列没有任何标识符号则意味着相应的模型和“cv. rf. v3”模型之间有显著差异的可能性低于95%（检查图例以理解符号的含义）。加号（“+”）意味着相应模型的平均性能估计指标显著高于模型“cv. rf. v3”。由于好的模型对应较低的NMSE值，所以该模型的性能比模型“cv. rf. v3”差。减号（“-”）的含义相反。

从输出结果可以确认，随机森林不同版本之间的差异在统计上通常不显著。与其他模型相比，在大部分情况下，随机森林模型有显著的优势。

可以对在其他海藻上有最优性能的模型进行如上类似的分析，只要在函数compAnalysis()的参数against和datasets上取不同的值即可。

84～90

模型选择和模型评价的参考文献

在不同的模型间进行比较和选择一直是许多研究的主题。其中，我们建议参考Dietterich（1998）、Provost等（1998）、Nemenyi（1969）和Demsar（2006）的书。

关于组合学习方法，也有大量的文献。我们重点推荐（Breiman，1996）关于bagging的书，（Freund and Shapire，1996；Shapire，1990）关于boosting的书。Dietterich（2000）是一篇很好的关于这些主题的一个综述。

2.8 预测 7 类海藻的频率

本节将学习如何给出 7 类海藻 140 个测试样本的频率预测值。2.7 节描述了如何选择预测值的最佳模型，给出了通过交叉验证实验过程来得到 7 个预测任务的预测模型的无偏 NMSE 估计方法。

本章数据挖掘案例的主要目的是得到 140 个水样的 7 个海藻的频率值预测。每一个预测任务都采用交叉验证过程给出的最佳模型进行预测，这个最佳模型将是 2.7 节中调用函数 bestScores() 所显示的模型之一，也就是从模型"cv. rf. v3"、"cv. rf. v2"、"cv. rf. v1"或"cv. rpart. v3"中选择最佳的。

下面应用所有可得的训练数据来构建模型，并将得到的模型应用到测试数据集。注意，为了简单，在构建回归树时，采用 k 近邻填补法填充数据框 clean. algae 的 NA 值。这样就避免回归树采用它自身的缺失值处理方法。随机森林本身没有缺失值处理方法，因此把数据框 clean. algae 作为它的训练集数据。下面的代码可以同时获得所有 7 个模型：

```
> bestModelsNames <- sapply(bestScores(res.all),
+                           function(x) x['nmse','system'])
> learners <- c(rf='randomForest',rpart='rpartXse')
> funcs <- learners[sapply(strsplit(bestModelsNames,'\\.'),
+                         function(x) x[2])]
> parSetts <- lapply(bestModelsNames,
+                    function(x) getVariant(x,res.all)@pars)
> bestModels <- list()
> for(a in 1:7) {
+   form <- as.formula(paste(names(clean.algae)[11+a],'~ .'))
+   bestModels[[a]] <- do.call(funcs[a],
+           c(list(form,clean.algae[,c(1:11,11+a)]),parSetts[[a]]))
+ }
```

上面的代码中得到了一个向量，其元素为每一个预测任务的最优模型。可以得到保存最优模型向量 funcs 中的相应最优模型的 R 函数名。可以通过函数 strsplit() 来提取模型的名称，这一步骤需要用到复杂的函数复合。为了理解整个过程，可以把以上获取函数名的这个函数复合分开来执行。每一个最优模型的参数赋给列表 parSetts。函数 getVariant() 给出相应给定名称的模型，这个函数的返回（对象）值是模型类对象。这些对象有不同的"属性"，其中一个是名为 pars 的属性，它包含模型参数列表。对象的属性可以用 R 的操作符"@"来访问。最后，得到最优模型并把它们赋给列表 bestModels。对于每一个海藻，如前面一样构建公式，然后通过函数 do. call() 调用适当设置的相应的 R 函数。函数 do. call() 可以调用任何函数，它的第一个参数是作为字符串的函数名，第二个参数是包含调用函数所需参数的一个列表。执行 do. call() 函数后，得到预测相应 7 个海藻类频率的最优模型，然后就可以应用这些模型对测试集进行预测[⊖]。

本书提供的 R 添加包的数据框 test. algae 含有 140 个测试水样，这个测试集中也含有缺失值。因此，第一步是用前面的方法来填补缺失值。首先尝试对测试集数据框应用函数 knnImputation() 来填补缺失值，该方法可行，但问题是这里有些违背预测模型的黄金法则"不要应用测试集中的任何信息来建立预测模型"。因此，如果直接在测试集上应用 knnImputation() 函数，它将应用测试集数据寻找 10 个最近邻值，并以此来填补缺失值。应用目标变量信息来建立模型是绝对错误的，尽管我们没有犯该错误，但是可以避免应用测试集数据来填补缺失值。方法是应用训练集数据中的 10 个最近邻元素来填补测试集中的缺失值。这样就更加贴合实际，也更正确。实际

⊖ 警告：如果打印对象 bestModels，输出结果可能充满整个屏幕。

上，我们可能是依次获取水样，一次得到一个水样。函数 knnImputation( ) 有一个特殊的参数，可以用训练集数据来填补测试集中的缺失值。用法如下：

```
> clean.test.algae <- knnImputation(test.algae, k = 10, distData = algae[,
+     1:11])
```

91 ~ 92

可以设置函数 knnImputation( ) 的参数 distData 来指定特定的数据集，从该数据集中找到测试集数据框 test. algae 中有缺失数据案例的 10 个最近邻值。注意，由于测试数据集没有目标变量的信息，因此在数据集 algae 中忽略了目标变量。

下面就可以获取整个测试数据集的预测值矩阵，代码如下：

```
> preds <- matrix(ncol=7,nrow=140)
> for(i in 1:nrow(clean.test.algae))
+   preds[i,] <- sapply(1:7,
+                       function(x)
+                         predict(bestModels[[x]],clean.test.algae[i,])
+                      )
```

在上面的简单代码中，需要的 7 × 140 个预测值存储在矩阵 preds 中。在这个预测问题中，由于测试集目标变量值实际上是已知的，所以可以把预测值和真实值进行比较，据此可以评价预测结果的质量。测试集数据的真实值在本书 R 添加包的数据框 algae. sols 中。下面的代码计算模型的 NMSE 值：

```
> avg.preds <- apply(algae[,12:18],2,mean)
> apply( ((algae.sols-preds)^2),           2,mean) /
+ apply( (scale(algae.sols,avg.preds,F)^2),2,mean)

       a1        a2        a3        a4        a5        a6        a7
0.4650380 0.8743948 0.7798143 0.7329075 0.7308526 0.8281238 1.0000000
```

首先得到计算 NMSE 值时需要用到的基准模型的预测值，这里是目标变量平均值的预测。基准预测由一行代码完成，它初看起来有点复杂，一旦你理解了该代码，你将惊异它的简洁性。函数 scale( ) 用来标准化数据集，如果第三个参数不是 FALSE，它从第一个参数中减去第二个参数，然后除以第三个参数，如上面的代码所示。在上面的例子中，我们用该函数从矩阵数据的每一行中减去一个向量（即所有 7 个藻类的平均目标变量值）。

得到的结果和前面交叉验证的估计结果相一致。它也再次确认很难得到海藻 7 的较好预测，而其他海藻的估计结果则相对较好，海藻 1 的估计结果最佳。

总之，通过适当的模型选择过程，就可以得到这些预测问题的恰当的分数。

93

2.9 小结

作为本书的第一个学习案例，本章主要目的是让读者熟悉 R 软件。据此，从数据挖掘的标准而言，这里选了一个较小的问题。本章描述了如何在 R 中进行一些最基本的数据分析任务。

如果你要了解基于本章数据的国际数据分析比赛的内容，可以浏览比赛的网站[⊖]，或者阅读一些有关获奖答案的文章（Bontempi et al. , 1999；Chan，1999；Devogelaere et al. , 1999；Torgo，1999b），然后比较文章作者所用的数据分析策略。

就数据挖掘而言，本案例描述了下列内容：

- 数据可视化
- 描述性统计分析

⊖ http：//www. erudit. de/erudit/competitions/ic-qq/.

- 处理缺失值的策略
- 回归分析
- 回归分析的评价指标
- 多元线性回归
- 回归树
- 通过 k 折交互验证来进行模型选择和比较
- 模型组合和随机森林

经过上面的学习，希望读者熟悉交互式应用 R 的方法，熟悉 R 的一些特性。也就是说，应该学习下列 R 的技能：

- 读入文本数据文件
- 如何得到数据集的描述性统计量
- 基本的数据可视化方法
- 如何处理有缺失值的数据
- 如何构建回归模型
- 如何应用模型得到测试集的预测值

94

本书接下来的案例研究将给出上面更多的细节和数据挖掘技巧。



预测股票市场收益

本书的第二个案例是对数据挖掘技术更深一层的使用。我们将分析把数据挖掘工具和技术应用到具体商业问题中所面临的困难。这里以建立自动股票交易系统这一具体问题为例。基于股票的每日交易数据，首先建立预测模型，然后据此分析如何建立股票交易系统。我们的目标是预测 S&P 500（标准普尔 500）股票指数的未来收益，为此首先建立了几个预测模型，然后这些模型结合给定的交易策略产生市场上的交易决策（即买卖信号）。本章主要讲解几个新的数据挖掘问题，它们是：1）如何使用 R 来分析存储在数据库中的数据；2）如何对具有时间顺序的观测值（即时间序列）进行预测；3）把模型预测结果转化为现实应用中的决策和行动。

3.1 问题描述与目标

对数据挖掘而言股票市场交易是个具有巨大潜力的应用领域。事实上，由于大量历史数据的存在，人工对这些数据进行检测是很困难的，而数据挖掘技术对大数据有先天的优势。另一方面，有学者声称，市场在价格调整上适应之快，以至于根本没有空间可以得到稳定的收益。这就是有名的*有效市场假设*。这个理论先后被一些更宽松的版本所取代，即由于短暂的市场无效，市场还是有一些交易机会空间的。

股票交易的总体目标是维持一个基于买卖订单的股票组合。长期目标就是从这些股票交易中获取尽可能多的利润。本章对股票组合简化，只“交易”一只单一的证券，这里采用股票市场指数——标准普尔指数。对于给定的证券和初始资金，我们将尝试通过交易行为（买入、卖出、持有），在未来一段测试期使利润最大化。应用数据挖掘技术得到结果给出信号，然后据此作为决策的基础来制定交易策略。在该过程中，我们应用标准普尔 500 指数的历史数据来预测未来的指数变化。因此我们的预测模型将包含进一个交易系统中，该交易系统应用模型的预测结果来生成决策。总体的评估标准就是该交易系统的性能，即该交易系统的交易所产生的利润或者损失，以及对投资者有意义的一些其他统计指标。因此，我们的主要评价指标是应用数据挖掘过程发现的知识来进行交易所产生的结果，而不是在该过程中所开发的模型的预测准确性。

3.2 可用的数据

在我们的案例中将关注交易标准普尔 500 指数。这个指数的日常数据在很多地方都可以获得，比如 Yahoo 财经网站[⊖]。

我们要用到的数据可以在本书的添加包中得到。同时，为了说明 R 的功能，我们会给出获取该数据的其他方式。另外，这些获取数据的方式可以让你把本章学到的知识应用到最近的数据中，而不是仅仅限于截止本书写作时打包的数据。

为了通过本书的 R 程序包得到这些数据，可以在 R 中输入：

⊖ http://finance.yahoo.com.

```
> library(DMwR)
> data(GSPC)
```

只有先前没有在当前的 R 会话中执行第一条指令的情况下，第一条指令才需要输入。第二条指令载入一个 GSPC ㊀对象，该对象是 xts 类的。我们将在 3.2.1 节讲解 xts 类对象。现在，只要把它作为一个矩阵或者数据框来操作即可（比如 head(GSPC)）。

在本书的网站㊁中，可以找到两种格式的数据。第一种是逗号分隔值（Comma Separated Value，CSV）文件，它可以被读到 R 中（如第 2 章所述）。另一种格式是 MySQL 数据库导出文件，可以用它在 MySQL 中生成一个存放 S&P 500 的数据库。我们将说明如何在 R 中导入这两种格式的数据。具体采用哪种格式的数据，完全取决于你自己，或者你也可以采用最简单的方式——直接应用本书提供的 R 添加包中的数据。本章的其他部分（导入数据之后的分析部分）与你所应用的存储数据的方式无关。

96

为了完整，我们也给出了另一种将数据读到 R 中的方法，即直接从数据网站上下载需要的数据。注意，如果选择这种方式，那么你所应用的数据集会比本章中的数据集大很多。

不管采用哪种方式获取数据，股票的日交易数据应该包括下面几个属性：

- 交易日期
- 当日开盘价
- 当日最高价
- 当日最低价
- 当日收盘价
- 当日成交量
- 当日调整后的收盘价㊂

3.2.1 在 R 中处理与时间有关的数据

这个案例中用到的数据和时间有关，即每个观测值有一个时间标签。该类数据常称为时间序列数据。由于每个观测值都有一个给定的时间标签，所以时间序列数据的重要特征是观测值之间的先后顺序很重要。一般来说，时间序列就是随机变量 Y 的一组有序的观测值，即

$$y_1, y_2, \cdots, y_{t-1}, y_t, y_{t+1}, \cdots, y_n \tag{3-1}$$

这里 y_t 是时间序列变量 Y 在时间 t 的观测值。

时间序列分析的主要目的根据变量过去的观测值 y_1，y_2，…，y_{t-1}，y_t 来构造一个模型，据此可以对时间序列未来的取值进行预测，即预测 y_{t+1}，…，y_n。

在本章的股票数据案例中，由于我们在同一个时间点上观测了多个变量，它们各自是 Open、High、Low、Close、Volume 和 AdjClose㊃，所以该类型的时间序列常称为多元时间序列。

97

R 有多个添加包用于对这类数据进行分析，这些包提供了特殊的类来存储不同类型的时间序列数据。而且，R 有许多函数用于这些不同类型的时间序列数据，例如特殊的绘图函数等。

对于处理时间序列数据，R 中最灵活的添加包有 zoo（Zeileis and Grothendieck，2005）和 xts（Ryan and Ulrich，2010）。这两个包提供类似的功能，但是 xts 包提供了用 ISO 8601 时间字符串来获取数据子集等更多的方法。在技术上，xts 类扩充了 zoo 类，每个 xts 类都同时是一个 zoo 类，

㊀ 在我们下载行情数据的雅虎财经网站，标准普尔 500 指数的股票编号为 GSPC。

㊁ http://www.liaad.up.pt/~ltorgo/DataMiningWithR.

㊂ 该价格是调整了股票分割、分红、配股等之后的价格。

㊃ 事实上，更严格地说，这里应该只有两个时间序列（价格 Price 和成交量 Volume），因为所有的报价是同一个变量（价格 Price）在一天中不同时间的取样。

因此 zoo 类对象的所有方法都可以应用到 xts 类对象。本章的分析主要是应用 xts 对象。注意，zoo 和 xts 都是 R 的附加包，它们在 R 的基本安装中是没有的，在应用之前需要先下载并安装它们（参见 1.2.1 节）。

下面举例说明如何创建和应用 xts 对象。

```
> library(xts)
> x1 <- xts(rnorm(100), seq(as.POSIXct("2000-01-01"), len = 100,
+     by = "day"))
> x1[1:5]

                  [,1]
2000-01-01  0.82029230
2000-01-02  0.99165376
2000-01-03  0.05829894
2000-01-04 -0.01566194
2000-01-05  2.02990349

> x2 <- xts(rnorm(100), seq(as.POSIXct("2000-01-01 13:00"),
+     len = 100, by = "min"))
> x2[1:4]

                         [,1]
2000-01-01 13:00:00 1.5638390
2000-01-01 13:01:00 0.7876171
2000-01-01 13:02:00 1.0860185
2000-01-01 13:03:00 1.2332406

> x3 <- xts(rnorm(3), as.Date(c("2005-01-01", "2005-01-10",
+     "2005-01-12")))
> x3

                 [,1]
2005-01-01 -0.6733936
2005-01-10 -0.7392344
2005-01-12 -1.2165554
```

函数 xts( ) 的第一个参数接收时间序列数据。该数据可以是一个向量，或者如果时间序列是多元时间序列，该参数也可以是一个矩阵[⊖]。在第二种情况下，矩阵的每一列解释为一个变量在不同时间点（即每一行）上的抽样值。第二个参数是时间标签，它可以是 R 时间类的任何一种形式。在上面的示例中，用到了 R 表示时间信息的两个最常用的时间类：POSIXct 类（或 POSIXlt 类）和 Date 类。有许多和这些类相关联的函数可以用来操作这些类中的时间信息，详细参见 R 的帮助文档。例如，在第 1 章中我们用 seq( ) 函数来生成一系列数，这里我们用该函数[⊜]生成一个基于时间的序列。 98

在上面的例子中，如果去掉时间标签，那么这些时间序列对象可以像“正常”对象一样进行索引（标准的向量子集）。我们经常需要基于与时间有关的条件来获取这些序列的子集。采用 xts 对象可以有多种方式实现它，举例如下：

```
> x1[as.POSIXct("2000-01-04")]

                  [,1]
2000-01-04 -0.01566194
```

⊖ 这意味着不能在 xts 中有像数据框那样的混合类型数据。

⊜ 事实上，这里是应用于类 POSIXt 的泛型函数 seq( ) 的一个特殊方法。可以输入“?seq. POSIXt”获得该函数的更多信息。

```
> x1["2000-01-05"]
                 [,1]
2000-01-05 2.029903

> x1["20000105"]
                 [,1]
2000-01-05 2.029903

> x1["2000-04"]
                          [,1]
2000-04-01 01:00:00  0.2379293
2000-04-02 01:00:00 -0.1005608
2000-04-03 01:00:00  1.2982820
2000-04-04 01:00:00 -0.1454789
2000-04-05 01:00:00  1.0436033
2000-04-06 01:00:00 -0.3782062
2000-04-07 01:00:00 -1.4501869
2000-04-08 01:00:00 -1.4123785
2000-04-09 01:00:00  0.7864352

> x1["2000-03-27/"]
                           [,1]
2000-03-27 01:00:00  0.10430346
2000-03-28 01:00:00 -0.53476341
2000-03-29 01:00:00  0.96020129
2000-03-30 01:00:00  0.01450541
2000-03-31 01:00:00 -0.29507179
2000-04-01 01:00:00  0.23792935
2000-04-02 01:00:00 -0.10056077
2000-04-03 01:00:00  1.29828201
2000-04-04 01:00:00 -0.14547894
2000-04-05 01:00:00  1.04360327
2000-04-06 01:00:00 -0.37820617
2000-04-07 01:00:00 -1.45018695
2000-04-08 01:00:00 -1.41237847
2000-04-09 01:00:00  0.78643516

> x1["2000-02-26/2000-03-03"]
                  [,1]
2000-02-26  1.77472194
2000-02-27 -0.49498043
2000-02-28  0.78994304
2000-02-29  0.21743473
2000-03-01  0.54130752
2000-03-02 -0.02972957
2000-03-03  0.49330270

> x1["/20000103"]
                 [,1]
2000-01-01 0.82029230
2000-01-02 0.99165376
2000-01-03 0.05829894
```



第一个命令中的参数是一个具体的时间值对象，它和创建 x1 对象的函数的第二个参数的对象的类是一样的。其他的例子展示了 xts 包中强大的子集索引功能，它是 xts 包优于 R 中的其他时间序列包的地方之一。这里的子集索引把时间标签作为字符串，其一般格式为：CCYY－MM－DD HH：MM：SS[.s]。上述格式的不同段之间的分隔符可以省略，该格式的某些部分可以忽略以用于表示一个时间标签集合。另外，“/”符号可以用于上述格式的开始或结尾，用以表示某个时间段。“/”符号在结尾表示以该时间开始的时间段，“/”符号在开始表示以该时间结束的时间段。

多元时间序列可以用类似的方式建立，例如：

```
> mts.vals <- matrix(round(rnorm(25),2),5,5)
> colnames(mts.vals) <- paste('ts',1:5,sep='')
> mts <- xts(mts.vals,as.POSIXct(c('2003-01-01','2003-01-04',
+                     '2003-01-05','2003-01-06','2003-02-16')))
> mts

            ts1   ts2   ts3   ts4   ts5
2003-01-01 0.96 -0.16 -1.03  0.17  0.62
2003-01-04 0.10  1.64 -0.83 -0.55  0.49
2003-01-05 0.38  0.03 -0.09 -0.64  1.37
2003-01-06 0.73  0.98 -0.66  0.09 -0.89
2003-02-16 2.68  0.10  1.44  1.37 -1.37

> mts["2003-01",c("ts2","ts5")]

             ts2   ts5
2003-01-01 -0.16  0.62
2003-01-04  1.64  0.49
2003-01-05  0.03  1.37
2003-01-06  0.98 -0.89
```

函数 index() 和 time() 可以用来“获取”任意 xts 对象的时间标签信息，coredata() 函数用来获取时间序列的观测值。例如，

```
> index(mts)

[1] "2003-01-01 WET" "2003-01-04 WET" "2003-01-05 WET" "2003-01-06 WET"
[5] "2003-02-16 WET"

> coredata(mts)

      ts1   ts2   ts3   ts4   ts5
[1,] 0.96 -0.16 -1.03  0.17  0.62
[2,] 0.10  1.64 -0.83 -0.55  0.49
[3,] 0.38  0.03 -0.09 -0.64  1.37
[4,] 0.73  0.98 -0.66  0.09 -0.89
[5,] 2.68  0.10  1.44  1.37 -1.37
```

总之，xts 对象可以用于存储带有不规则时间标签的多元时间序列，它足以用来存储股票交易数据，并能提供强大的子集索引功能。

3.2.2 从 CSV 文件读取数据

如前文所述，本书网站有本章案例的各种数据格式。如果你决定用 CSV 数据文件，需要先下载该数据文件，它的前面几行为：

```
"Index"    "Open" "High" "Low" "Close" "Volume" "AdjClose"
1970-01-02 92.06 93.54 91.79 93    8050000 93
1970-01-05 93    94.25 92.53 93.46 11490000 93.46
1970-01-06 93.46 93.81 92.13 92.82 11460000 92.82
1970-01-07 92.82 93.38 91.93 92.63 10010000 92.63
```

99 ~ 101



```
1970-01-08 92.63  93.47  91.99 92.68   10670000 92.68
1970-01-09 92.68  93.25  91.82 92.4    9380000  92.4
1970-01-12 92.4   92.67  91.2  91.7    8900000  91.7
```

假如你已经下载了该文件，并且用文件名“sp500.csv”保存在当前运行 R 的目录中，那么可以把数据载入到 R 中并用之创建一个 xts 对象，代码如下所示：

```
> GSPC <- as.xts(read.zoo("sp500.csv", header = T))
```

假如 CSV 数据文件的第一列为时间标签，则 zoo[⊖]里的 read.zoo() 函数可以读取该 CSV 数据文件并把数据转换为 zoo 对象。函数 as.xts() 把读取的结果对象转换为 xts 对象。

3.2.3 从网站上获取数据

获取 S&P 500 指数的另一种方法是使用 Yahoo 财经网站提供的免费服务，它允许以 CSV 数据文件格式下载股票报价。R 的 tseries 添加包[⊜]有函数 get.hist.quote()，该函数可以用来下载报价数据到一个 zoo 对象。下面就是一个使用该函数获取 S&P 500 报价的例子。

```
> library(tseries)
> GSPC <- as.xts(get.hist.quote("^GSPC",start="1970-01-02",
         quote=c("Open", "High", "Low", "Close","Volume","AdjClose")))

...
...

> head(GSPC)

            Open  High   Low Close   Volume AdjClose
1970-01-02 92.06 93.54 91.79 93.00  8050000    93.00
1970-01-05 93.00 94.25 92.53 93.46 11490000    93.46
1970-01-06 93.46 93.81 92.13 92.82 11460000    92.82
1970-01-07 92.82 93.38 91.93 92.63 10010000    92.63
1970-01-08 92.63 93.47 91.99 92.68 10670000    92.68
1970-01-09 92.68 93.25 91.82 92.40  9380000    92.40
```

由于函数 get.hist.quote() 返回一个 zoo 类的对象，所以我们用函数 as.xts() 把该对象转换为一个 xts 对象。注意，如果运行该命令，我们将得到比本书 R 包中所提供的数据更大的一个数据集。如果想确保本章后面的命令得到和本书中同样的结果，应该应用以下的代码：

102

```
> GSPC <- as.xts(get.hist.quote("^GSPC",
         start="1970-01-02",end='2009-09-15',
         quote=c("Open", "High", "Low", "Close","Volume","AdjClose")))
```

这里，“2009-09-15”是本书添加包中的 GSPC 对象的最后一个记录交易日期。

另外一种通过网络获取数据的方法（我们后面会看到，这不是唯一的方法）是应用 quantmod（Ryan，2009）包中的函数 getSymbols()。quantmod 包是 R 的添加包，在应用之前确保该包已经安装。该包提供了与分析金融数据有关的功能，本章将应用这些功能。函数 getSymbols() 和该包中的其他函数一起提供了获取不同来源交易数据的相对简单、但是功能强大的方法。下面举例说明该函数的用法：

```
> library(quantmod)
> getSymbols("^GSPC")
```

⊖ 你可能要问这里为什么不调用函数 library() 来载入添加包 zoo。原因是 xts 添加包依赖于添加包 zoo，当载入 xts 包时就同时载入了 zoo 包。

⊜ 需要首先安装该添加包。

函数 getSymbols( ) 的第一个参数是一个符号集合，该函数将从不同网站或本地数据库中提取这些符号所对应的交易数据，然后默认返回与符号同名的 xts 对象[⊖]。这些操作都是在 R 工作环境下自动完成的。你可以证实返回的对象数据与本书包中数据的时间不是完全相同，列的名称也有不同。可以通过下列代码修正：

```
> getSymbols("^GSPC", from = "1970-01-01", to = "2009-09-15")
> colnames(GSPC) <- c("Open", "High", "Low", "Close", "Volume",
+     "AdjClose")
```

事实上，在 quantmod 包提供的框架中，可以有多个符号来自不同的相关数据源，每个都有各自的参数。这些设置都可以在 R 会话开始时用函数 setSymbolsLookup( ) 来指定，如下所示：

```
> setSymbolLookup(IBM=list(name='IBM',src='yahoo'),
+                 USDEUR=list(name='USD/EUR',src='oanda))
> getSymbols(c('IBM','USDEUR'))

> head(IBM)

           IBM.Open IBM.High IBM.Low IBM.Close IBM.Volume IBM.Adjusted
2007-01-03    97.18    98.40   96.26     97.27    9196800        92.01
2007-01-04    97.25    98.79   96.88     98.31   10524500        93.00
2007-01-05    97.60    97.95   96.91     97.42    7221300        92.16
2007-01-08    98.50    99.50   98.35     98.90   10340000        93.56
2007-01-09    99.08   100.33   99.07    100.07   11108200        94.66
2007-01-10    98.50    99.05   97.93     98.89    8744800        93.55

> head(USDEUR)

           USDEUR
2009-01-01 0.7123
2009-01-02 0.7159
2009-01-03 0.7183
2009-01-04 0.7187
2009-01-05 0.7188
2009-01-06 0.7271
```

在上面的代码中，我们指定了两个符号所对应的获取行情数据的网站：IBM 数从 Yahoo 网站获取；USDEUR（美元和欧元）的汇率数据从 Oanda 网站[⊖]获取。

这些是通过函数 getSymbolsLookup( ) 完成的。在当前 R 会话中，当调用 getSymbols( ) 函数获取以上设置的标识符数据时，将应用由函数 getSymbolsLookup( ) 所做的相关设置。函数 saveSymbolLookup( ) 和 loadSymbolLookup( ) 分别用来在不同的 R 会话中保存和载入这些设置。如果需要这些函数的更多例子和详细解释，请参照 R 中这些函数的帮助文档。

3.2.4 从 MySQL 数据库读取数据

另一种存储数据的形式是在 MySQL 数据库里。在本书的网站上，有一个包含 SQL 语句的文件，它可以下载并用 MySQL 来执行，从而上载 S&P 500 数据到一个数据库表里。有关应用和创建 MySQL 数据库的内容可以参阅 1.3 节。

创建一个数据库来存储数据后，就可以准备执行 SQL 语句了。假设该文件和输入的 MySQL 命令在同一个目录下，同时，创建的数据库名为 Quotes。登录到 MySQL，然后输入：

```
mysql> use Quotes;
mysql> source sp500.sql;
```

⊖ 需要首先安装添加包。

⊖ http://www.oanda.com

文件“sp500.sql”中的 SQL 语句将创建一个名为 gspc 的数据表，并把这个案例中用到的数据插入到表中。你可以在 MySQL 中执行以下语句来确认数据表已正常建立：

```
mysql> show tables;
+------------------+
| Tables_in_Quotes |
+------------------+
| gspc             |
+------------------+
1 row in set (0.00 sec)

mysql> select * from gspc;
```

最后一条 SQL 语句会输出大量数据，即标准普尔 500 的报价数据。如果你想要限制输出的数据，你可以在以上命令句末尾添加 limit 10。

在 R 中有两种方式和数据库进行通信。一种方式是基于 ODBC 协议，另一种是基于 DBI 包提供的通用接口和每个数据库管理系统（DBMS）专有的包。

如果你决定应用 ODBC 协议，那么你要确认可以应用该协议和数据库通信，这可能需要在数据库端安装相关的驱动程序。在 R 端，需要的是安装 RODBC 包。

DBI 包实现了一系列的数据库接口函数，这些函数独立于实际上存储数据的数据库服务器。只需要在最初和数据库建立连接时指出用户将应用哪一个通信接口。当改变数据库时，你只要改变一条指令即可（该指令指出你希望通信的数据库）。为了获取这种独立性，用户需要安装其他的 R 包来处理不同数据库的通信细节。对于常用的数据库，R 有许多专有的数据库包。这里，为了和存储于服务器上的 MySQL 数据库通信，我们需要 RMySQL 包（James and DebRoy，2009）。

3.2.4.1　在 Windows 系统上加载数据到 R 中

如果在 Windows 系统上运行 R，那么无论 MySQL 数据库服务器是在该台计算机上，还是在另一台计算机上（可以是其他操作系统），从 R 连接该数据库的最简便方法是通过 ODBC（开放数据库互连）协议。为了在 R 中使用这个协议，需要安装 RODBC 添加包。

第一次使用 ODBC 协议连接任何 MySQL 数据库前，需要一些额外的步骤，即你需要在 Windows 系统上安装 MySQL ODBC 驱动程序，它称为“myodbc”，可以从 MySQL 网站上下载。这个步骤只在第一次使用 ODBC 来连接 MySQL 数据库时需要。安装好这个驱动程序后，就可以创建到 MySQL 数据库的连接，这个数据库可以是在你的计算机上或者其他可以从网络中访问到的计算机上。根据 ODBC 协议，创建的每个数据库连接都有一个名字（根据 ODBC 的术语，该连接称为*数据源名*）。这个名字用来从 R 来访问 MySQL 数据库。

103 ~ 105

在 Windows 计算机上创建一个 ODBC 连接，需要一个称为“ODBC 数据源”的程序，可以在 Windows 系统的控制面板上找到该程序。运行这个程序后，需要用 MySQL ODBC 驱动程序（myodbc）创建一个新的用户数据源。在创建数据源的过程中，有几个问题会问到，比如 MySQL 服务器地址（如果是本机，则为 localhost；如果是远程主机，则形如 myserver.xpto.pt）；需要建立连接的数据库名；还有该连接的名称。一旦完成这个过程后，就可以从 R 连接 MySQL 数据库了。

以下 R 代码建立了一个到 Quotes 数据库的连接，并且把 S&P 500 数据加载到 R。

```
> library(RODBC)
> ch <- odbcConnect("QuotesDSN",uid="myusername",pwd="mypassword")
> allQuotes <- sqlFetch(ch,"gspc")
> GSPC <- xts(allQuotes[,-1],order.by=as.Date(allQuotes[,1]))
> head(GSPC)

            Open  High   Low Close  Volume AdjClose
1970-01-02 92.06 93.54 91.79 93.00 8050000    93.00
```

```
1970-01-05 93.00 94.25 92.53 93.46 11490000     93.46
1970-01-06 93.46 93.81 92.13 92.82 11460000     92.82
1970-01-07 92.82 93.38 91.93 92.63 10010000     92.63
1970-01-08 92.63 93.47 91.99 92.68 10670000     92.68
1970-01-09 92.68 93.25 91.82 92.40  9380000     92.40

> odbcClose(ch)
```

加载 RODBC 包后，用先前创建的 DSN [1]使用 odbcConnect( ) 函数建立与数据库的连接。然后用 sqlFetch( ) 函数（它包含了表格的所有行，并作为数据框对象返回）查询数据表。下一个步骤就是利用日期信息和交易数据从这个数据框创建一个 xts 对象。最后，用 odbcClose( ) 函数关闭与数据库的连接。

当使用特别大的数据库时，这里要注意的是：如果查询产生的结果太多，以至于计算机内存存储不下，那么你就不得不用其他的策略。你可以尝试对数据进行分块处理，这需要在函数 sqlFetch( ) 和 sqlFetchMore( ) 中设置 max 参数。 106

也可以考虑应用其他方法，例如考虑 R 中提供高性能和并行计算功能[2]的包，ff 包（Aoler et al. , 2010）。

3.2.4.2　在 Linux 系统上加载数据

如果在类 UNIX 系统上运行 R，那么连接 MySQL 数据库最容易的途径是通过 DBI 包与 RMySQL 包的结合。这里，ODBC 协议对 Linux 系统也是可用的。有了 RMySQL 包后，你就不需要像应用 RODBC 包那样的准备步骤。安装完 RMySQL 包后，你就可以开始使用它，用法如下所示：

```
> library(DBI)
> library(RMySQL)
> drv <- dbDriver("MySQL")
> ch <- dbConnect(drv,dbname="Quotes","myusername","mypassword")
> allQuotes <- dbGetQuery(ch,"select * from gspc")
> GSPC <- xts(allQuotes[,-1],order.by=as.Date(allQuotes[,1]))
> head(GSPC)

            Open  High   Low Close   Volume AdjClose
1970-01-02 92.06 93.54 91.79 93.00  8050000    93.00
1970-01-05 93.00 94.25 92.53 93.46 11490000    93.46
1970-01-06 93.46 93.81 92.13 92.82 11460000    92.82
1970-01-07 92.82 93.38 91.93 92.63 10010000    92.63
1970-01-08 92.63 93.47 91.99 92.68 10670000    92.68
1970-01-09 92.68 93.25 91.82 92.40  9380000    92.40

> dbDisconnect(ch)

[1] TRUE

> dbUnloadDriver(drv)
```

加载包后，用函数 dbDriver( ) 和 dbConnect( ) 打开与数据库的连接。用函数 dbGetQuery 向数据库发送一个 SQL 查询，然后收到一个数据框结果。通常情况下，把结果转换为 xts 对象后，用 dbDisconnect( ) 和 dbUnloadDriver( )关闭数据库连接。

另外，可以应用 quantmod 包提供的函数来应用 MySQL 数据库中的数据。事实上，可以把

[1] 这里要用你在 Windows 控制面板中创建的数据源（DSN）和你的 MySQL 用户名和密码。

[2] http://cran. at. r-project. org/web/views/HighPerformanceComputing. html.

107 MySQL 数据库作为函数 getSmbols( ) 的一个数据源。示例如下：

```
> setSymbolLookup(GSPC=list(name='gspc',src='mysql',
+  db.fields=c('Index','Open','High','Low','Close','Volume','AdjClose'),
+  user='xpto',password='ypto',dbname='Quotes'))
> getSymbols('GSPC')

[1] "GSPC"
```

3.3 定义预测任务

一般来说，本章的目的是预测 S&P 500 指数的未来价格，据此可以及时进行交易以从中获利。有了分析目标后，就可以容易地定义模型需要预测的对象——预测价格时间序列的未来值。然而，即使面对这个简单的任务，我们也立即面临几个问题：1）采用哪一个每日价格；2）预测未来哪个时间。这些问题并不容易回答，它通常取决于在下达交易命令时如何应用预测结果。

3.3.1 预测什么

3.5 节将学习的交易策略假设我们可以预测未来几天的市场变化趋势。如果这个预测在未来被验证是正确的，那么基于该预测下达的交易指令将是获利的。

假设从交易方面看，我们认为价格变动超过 $p\%$ 时值得交易（即获利超过交易费用）。在这个假设下，我们需要预测模型来预测在未来 k 天中是否能够获得这个边际利润[⊖]。注意，在这 k 天中，实际上可以观察的价格可能高于这个比例和低于这个比例。这意味着，预测未来某个特定时间 $t+k$ 的报价可能不是最好的方法。事实上，我们需要预测的是在未来 k 天中价格总的动态变化，并不是预测某个特定时间的一个特定价格。例如，在时间 $t+k$ 的收盘价的变化可能比 $p\%$ 低得多，但是在它前面日期 t，…，$t+k$ 的价格变化可能远远大于 $p\%$。因此，我们事实上需要的是未来 k 天的总体价格趋势。

我们从报价数据计算一个变量，它可以作为未来 k 天的趋势指标值。这一指标值应与我们对 108 接下来的 k 天能够获得 $p\%$ 的价格变化的信心相关。在这个阶段，重要的是要注意，当我们提到 $p\%$ 的变化时，它的意思是高于或低于目前的价格。这里的想法是，正的变化将导致买入，而负的变化将触发卖出行动。这里我们给的指标把趋势作为一个单一的值，正值表示向上的趋势，负值表示价格向下的趋势。

假设每天的平均价格可以由下面公式来近似：

$$\overline{P}_i = \frac{C_i + H_i + L_i}{3} \tag{3-2}$$

其中，C_i、H_i 和 L_i 分别为第 i 天的收盘价、最高价和最低价。

设 V_i 代表未来 k 天的平均价格相对今天收盘价的百分比变化（通常称为算术收益）：

$$V_i = \left\{\frac{\overline{P}_{i+j} - C_i}{C_i}\right\}_{j=1}^{k} \tag{3-3}$$

我们把动态变化绝对值超过目标收益 $p\%$ 的变化进行累加作为一个指标变量 T：

$$T_i = \sum_v \{v \in V_i : v > p\% \vee v < -p\%\} \tag{3-4}$$

指标变量 T 用来找出在 k 天内，日平均价格明显高于目标变化的那些日期的变化之和。大的正 T 值意味着有几天的日平均报价高于今天收盘价的 $p\%$，这种情况表明有潜在的机会发出买入指令，因为有良好的预期价格会上涨。另一方面，大的负 T 值表明价格可能下降，可以进行卖出

⊖ 显然我们不希望等待若干年才能获得该边际利润。

行动。如果 T 值接近零，则可能是由于价格平稳波动或者价格涨跌互现，正的变化和负的变化互相抵消。

下面的函数实现这个简单的指标：

```
> T.ind <- function(quotes, tgt.margin = 0.025, n.days = 10) {
+     v <- apply(HLC(quotes), 1, mean)
+     r <- matrix(NA, ncol = n.days, nrow = NROW(quotes))
+     for (x in 1:n.days) r[, x] <- Next(Delt(v, k = x), x)
+     x <- apply(r, 1, function(x) sum(x[x > tgt.margin | x <
+         -tgt.margin]))
+     if (is.xts(quotes))
+         xts(x, time(quotes))
+     else x
+ }
```

根据式（3-2），函数首先计算平均价格。函数 HLC( ) 从价格对象中提取价格的最高价、最低价和收盘价。然后，计算未来 n. days 天相对当前收盘价的收益。函数 next( ) 按时间平移 109 一个时间序列（向前或向后）。Delt( ) 函数可用于计算价格序列的百分比收益或对数收益。最后，T. ind( ) 函数将绝对值大的收益相加，也就是说，收益超过目标变化收益，默认设置为 2.5%。

图 3-1 可以更好地理解指标 T 的性质，绘图的代码如下：

```
> candleChart(last(GSPC, "3 months"), theme = "white", TA = NULL)
> avgPrice <- function(p) apply(HLC(p), 1, mean)
> addAvgPrice <- newTA(FUN = avgPrice, col = 1, legend = "AvgPrice")
> addT.ind <- newTA(FUN = T.ind, col = "red", legend = "tgtRet")
> addAvgPrice(on = 1)
> addT.ind()
```

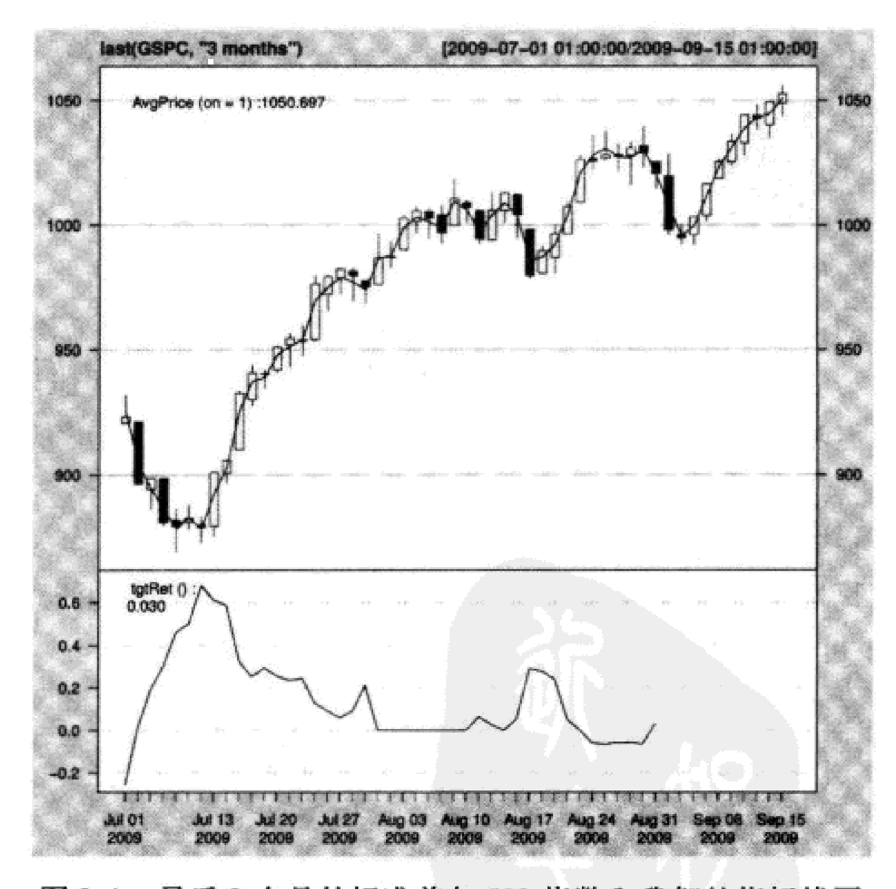


图 3-1　最后 3 个月的标准普尔 500 指数和我们的指标线图

函数 candleChart( ) 绘制股票价格的 K 线图。K 线图用一个彩色的框和竖直的柱条来代表每日报价情况。柱条代表当天的最高、最低价格，而框代表开盘价和收盘价。框的颜色用来表示框

的顶部所代表的价格（开盘价还是收盘价），即在一天中价格是下降（图 3-1 中为黑色，而交互式 R 输出的图形为橘色）还是上升（图 3-1 中为白色，在 R 中给出的为绿色）。我们在同一张 K 线图中，增加了另外两个指标：平均价格（和 K 线在同一张图上）和 *T* 指标（下面）。函数 newTA( ) 可用于绘制新的函数指标并加入到已有的 K 线图中。该函数的返回值是一个绘图函数[⊖]！这意味着可以像调用 R 的其他函数一样来调用对象 addT. ind 和对象 addAvgPrice。这由最后两条指令来完成，每一条指令把一个指标绘制到前面用 candleChart( ) 得到的 K 线图中。每条指令都增加一个指标让 candleChart( ) 函数产生初始图形。调用函数 addAvgPrice( ) 的参数为 1，这意味着该指标将被绘制到第一个图形窗口中，即前面得到的 K 线图中。调用函数 addT. ind( ) 时采用默认参数（而不是设为 1），这导致在 K 线图下绘制新的图形。由于指标 *T* 的量纲和 K 线图不同，所以在另一个图形中绘制 *T* 是合理的。

如图 3-1 所示，当有一系列的时期价格上升时，*T* 值达到了最大。显然，为了计算在时间 *i* 时指标 *T* 的取值，需要有未来 10 天的价格，因此这里不是用 *T* 来预测未来价格。预测未来价格不是指标 *T* 的目的，其目的是把未来观测到的价格变化概括为一个单一的值。

本案例的方法假设在时刻 *t* 的正确交易行为是和我们对未来 *k* 天价格的变化预期相关的。我们用指标 *T* 来描述未来价格的变化。如果 *T* 值高于一个给定的界限值，那么正确的交易信号应该是“买入”；如果 *T* 值低于一个给定的界限值，那么正确的交易信号将是“卖出”。在其他情况下，正确的交易方式将是什么也不做（即持有）。总之，我们想要预测时刻 *t* 的正确交易信号。对于历史价格数据，我们将计算每天的 *T* 值，并用给定的界限值，用上面给出的方法得到每一天的正确信号。

3.3.2 预测变量是什么

我们已经确定用指标（*T*）来总结接下来 *k* 天的价格时间序列行为。数据挖掘的目标是预测这种行为。试图预测未来金融市场行为背后的主要假设是通过观察市场过去的行为可以对未来做出预测。更确切地说，我们假设如果过去某些行为 *p* 之后是另一个行为 *f*，并且如果这一因果链经常发生，那么假设这一现象未来也会再次发生就是合理的。所以如果我们现在观察到现象 *p*，我们就可以预测下面将观察到现象 *f*。我们用指标 *T* 来近似表示未来的行为（*f*）。我们下面将给出如何描述近期的价格模式（即上面描述的现象 *p*）。这里不是使用一个单一的指标来描述价格的近期动态，而是使用多个指标来描述价格时间序列的不同属性，据此进行预测任务。

我们可以使用的描述过去最简单的信息类型是最近观测到的价格。通俗地说，这也是多个标准时间序列建模方法所应用的方法。这些方法对时间序列未来值和该序列过去 *q* 个观测值窗口之间的关系建模。我们除了描述当前时间序列的动态特性外，还会考虑最近一个窗口时期的价格特征。

技术指标反映价格时间序列特征的数值汇总。尽管应用这些指标作为决定交易的工具仍然具有争议性，但它们仍然可以提供价格时间序列动态特征的有意义的汇总。有大量的技术指标，在 R 的添加包 TTR 中可以找到这些指标（Ulrich，2009）。

这些技术指标通常用来获取价格序列的某些特征，例如价格是否波动太大、是否有某种特定趋势等。在本案例中，我们不会花大量时间来寻找适合我们预测任务的技术指标。而且，对于我们的案例，技术指标的选择仍然是一个研究课题。这通常称为变量选择问题，它用来从所有已知的输入变量中找到最适合建模的那个子集。变量选择方法通常分为两类：1）变量过滤器；2）变量封装。前者的变量选择独立于变量选择之后的模型构建阶段。它一般使用变量的某些统

⊖ 你可以通过输入 class(addT. ind) 来确认这一点，或者输入对象的名称来查看该对象的内容。

计特性（如相关）来选择用于建模的数据集。变量封装方法在变量选择过程中要包含后面要构建的模型信息。它是一个循环选择过程，在循环选择过程的每一步中，一组候选变量用于特定的模型并记录模型的相应结果。基于选择记录的结果，用某些搜寻操作确定一组新的候选输入变量子集，重复这个过程直到定义最终变量子集的收敛准则满足。

我们将采用简单的方法来选择用于模型的变量集合。这里通过一个具体的例子来演示如何选择变量。这里不是试图找出适合问题的最佳变量子集，这将需更多的时间和计算资源。首先我们给出一组初始输入变量，然后用一个方法来估计这些变量的重要性。基于它们的重要性，选择最适合问题的变量。

由于买入/卖出决策是在每天交易结束后，所以这里将集中于收盘价的分析。初始的输入变量由多个基于收盘价的过去收益构成。可以用下列公式来计算 h 天（算术）的收益[⊖]，或者百分比收益

112

$$R_{i-h} = \frac{C_i - C_{i-h}}{C_{i-h}} \tag{3-5}$$

这里 C_i 是第 i 天的收盘价。

在候选输入变量中包含了 h 从 1 ~ 10 的 10 个百分比收益。下一步，从 R 添加包 TTR 中选择有代表性的技术指标集合，即平均真实范围（Average True Range，ATR），该指标是衡量价格波动的指标；随机动量指数（Stochastic Momentum Index，SMI），该指标是一个动量指标；威尔斯－威尔德（Welles Wilder）定向运动指数（ADX）；Aroon 指标，该指标找出开始的趋势；布林带（Bollinger Bands）指标，它比较一段时间内价格的波动；蔡金波动（Chaikin Volatility）指标；收盘价位置价值（Close Location Value，CLV），该指标把收盘价和其交易范围相联系；阿姆氏简易波动指标（Ease of Movement Value，EMV）；MACD 指标；资金流向指数（Money Flow Index，MFI）；抛物线止损反转和波动性指标。以上指标的详细信息可以在 R 添加包 TTR 相应函数的帮助页面中找到。这些指标的大部分给出了多个值交易决策。如前所述，我们不打算使用这些指标交易。因此，我们对这些 TTR 函数的输出做了后续处理以获得一个单一值。以下函数实现了这个过程：

```
> myATR <- function(x) ATR(HLC(x))[, "atr"]
> mySMI <- function(x) SMI(HLC(x))[, "SMI"]
> myADX <- function(x) ADX(HLC(x))[, "ADX"]
> myAroon <- function(x) aroon(x[, c("High", "Low")])$oscillator
> myBB <- function(x) BBands(HLC(x))[, "pctB"]
> myChaikinVol <- function(x) Delt(chaikinVolatility(x[, c("High",
+     "Low")]))[, 1]
> myCLV <- function(x) EMA(CLV(HLC(x)))[, 1]
> myEMV <- function(x) EMV(x[, c("High", "Low")], x[, "Volume"])[,
+     2]
> myMACD <- function(x) MACD(Cl(x))[, 2]
> myMFI <- function(x) MFI(x[, c("High", "Low", "Close")],
+     x[, "Volume"])
> mySAR <- function(x) SAR(x[, c("High", "Close")])[, 1]
> myVolat <- function(x) volatility(OHLC(x), calc = "garman")[,
+     1]
```

上面描述的变量给出了预测指标 T 的未来取值的初始预测变量集合。下面应用变量选择方法把这 22 个预测变量进行精简。2.7 节应用了随机森林（Breiman，2001）方法预测海藻的频率。

⊖ 对数收益的定义为 $\log(C_i/C_{i-h})$。

随机森林也可以用于估计预测任务中变量的重要性。不严格地讲，它可以通过计算每个变量被[113]移除后随机森林误差的增加来估计变量的重要性。在某种意义上，这种方法和变量封装类似，它在选择变量过程中需要用到模型的信息。然而，这里不是通过循环过程来选择变量的，另外我们用其他的模型来预测 T，这就是说，这个变量选择过程没有对其他预测模型进行优化，因此它更像是变量过滤。

在本案例中，我们把数据集分成 2 个独立的子集：1）一个数据集用于构建交易系统；2）另一个用于测试。第一个数据集由 S&P 500 指数前 30 年的数据构成。剩下的数据（大约 9 年）用于最终交易系统的测试。基于此，为了避免出现有偏差的结果，最终的测试数据不用于变量选择过程。

用训练集数据构建随机森林模型，代码如下：

```
> data(GSPC)
> library(randomForest)
> data.model <- specifyModel(T.ind(GSPC) ~ Delt(Cl(GSPC),k=1:10) +
+        myATR(GSPC) + mySMI(GSPC) + myADX(GSPC) + myAroon(GSPC) +
+        myBB(GSPC)  + myChaikinVol(GSPC) + myCLV(GSPC) +
+        CMO(Cl(GSPC)) + EMA(Delt(Cl(GSPC))) + myEMV(GSPC) +
+        myVolat(GSPC)  + myMACD(GSPC) + myMFI(GSPC) + RSI(Cl(GSPC)) +
+        mySAR(GSPC) + runMean(Cl(GSPC)) + runSD(Cl(GSPC)))
> set.seed(1234)
> rf <- buildModel(data.model,method='randomForest',
+              training.per=c(start(GSPC),index(GSPC["1999-12-31"])),
+              ntree=50, importance=T)
```

上面的代码使用函数 specifyModel( ) 来设定并获取建模数据集。这个函数创建一个 quantmod 对象，它包含一个抽象模型（公式描述）的描述。该描述中所设定的数据可以有不同类型的来源，其中一些甚至目前还不在计算机的内存中。该函数将调用函数 getSymbols( ) 来处理数据源，并获取必要的数据。对于后面的建模阶段而言，这种指定数据和获取必要数据的方式是非常方便的。此外，对于来源为网络的数据符号，可以在后面把这里得到的对象（在我们例子中为 data. model）作为函数 getModelData( ) 的参数，用该函数来获取包含最新报价数据的对象。如果要使交易系统适应更新的报价信息，这种方法将十分便利。

函数 buildModel( ) 使用得到的模型规范，获得有相应数据的模型。通过参数 training. per，可以指定建立模型所用的数据（这里使用的是前 30 年的数据）。目前该函数包含了多个内置的[114]模型[1]，其中有随机森林。如果需要使用函数 buildModel( ) 中没有包含的模型，可以使用函数 modelData( ) 来获取数据，然后使用你喜爱的模型来建模。下例给出演示代码：

```
> ex.model <- specifyModel(T.ind(IBM) ~ Delt(Cl(IBM), k = 1:3))
> data <- modelData(ex.model, data.window = c("2009-01-01",
+     "2009-08-10"))
```

所获得的 data 对象是一个标准的 zoo 对象，它可以很容易地转换为矩阵或数据框，然后作为参数供任何建模函数使用，如下面的演示例子[2]：

```
> m <- myFavouriteModellingTool(ex.model@model.formula,
+                               as.data.frame(data))
```

注意，这里是如何表示模型公式的。建模中的“真实”公式不一定和函数 specifyModel( ) 参数中提供的公式完全一样。后者的公式用来获取数据，但“真正”的公式中可应用函数

---

[1] 可以查看帮助文档了解有哪些模型。

[2] 不要运行这段代码，这是一段“伪码”。

specifyModel( ) 得到的任何列和相应的名称。这些信息包含在函数 specifyModel( ) 生成的 quantmod 对象的 model. formula 属性中。

注意，上面的演示代码中所用到的股票（IBM），目前在内存中没有该股票的数据。specifyModel( ) 函数将应用函数 getSymbols( ) 透明地从网站获取该股票的价格数据。所有这一切对用户都是透明的，用户甚至可以在模型规范中包含来源不同的数据符号（例如，3.2.3 节中应用函数 setSymbolLookup( ) 的例子）。

回到变量选择问题，注意设置参数 importance = TRUE，这样随机森林将估计变量的重要性。对于回归问题，R 中的随机森林将估计两个可选变量的重要性分数。第一个分数是当依次删除每个变量时，随即森林错误增加的百分比。在每个变量被删除时，通过计算树在样本外数据上的均方误差的增加来估计该指标。该指标是对森林中所有树得到的结果进行并标准化得到的。第二个得分是与变量导致的结点杂质减少有关，当然它也是对所有树的平均值。我们将使用第一个得分，因为它是在随机森林（Breiman，2001）的原始文献中提到的方法。得到模型后，我们可以检查变量的重要性，方法如下：

```
> varImpPlot(rf@fitted.model, type = 1)
```

115

上面函数调用的结果在图 3-2 中给出。函数 varImpPlot( ) 的参数是随机森林和我们想绘制的得分（如果省略参数，将绘制两者）。泛型函数 buildModel( ) 返回作为结果产生的 quantmod 对象插槽（fitted. model）即为所获得的模型。泛型函数 buildModel( ) 返回的模型对象将是 quantmod 对象的一个属性。

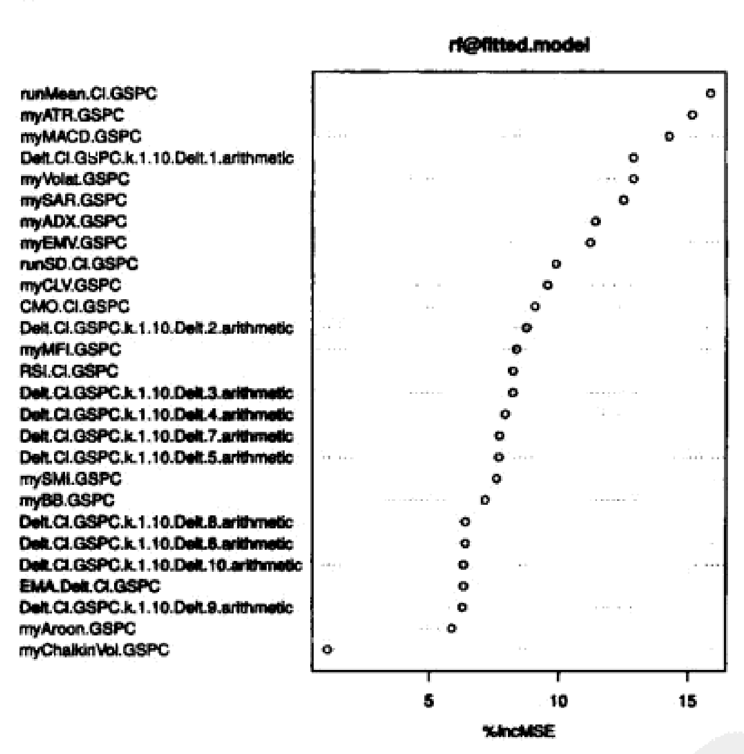


图 3-2 根据随机森林的变量重要性

现在，我们需要确定一个界限值来选择重要性评分高的变量子集。纵观图 3-2 的结果，因为这是一个简单的用随机森林来选择变量的例子，所以我们将使用界限值 10：

```
> imp <- importance(rf@fitted.model, type = 1)
> rownames(imp)[which(imp > 10)]

[1] "Delt.Cl.GSPC.k.1.10.Delt.1.arithmetic"
[2] "myATR.GSPC"
[3] "myADX.GSPC"
[4] "myEMV.GSPC"
```

```
[5] "myVolat.GSPC"
[6] "myMACD.GSPC"
[7] "mySAR.GSPC"
[8] "runMean.Cl.GSPC"
```

函数 importance( ) 将得到每个变量的具体重要性分数（这是第一个得分）。然后，我们用给116定的界限值来筛选用于建模的变量子集，我们将在我们的模拟尝试中使用这些变量。利用变量重要性的信息，我们可以得到最终用于建立模型的数据集：

```
> data.model <- specifyModel(T.ind(GSPC) ~ Delt(Cl(GSPC), k = 1) +
+     myATR(GSPC) + myADX(GSPC) + myEMV(GSPC) + myVolat(GSPC) +
+     myMACD(GSPC) + mySAR(GSPC) + runMean(Cl(GSPC)))
```

3.3.3 预测任务

在3.3.2节中，我们已经获得一个 quantmod 对象（data. model），该对象包含了用于建立预测模型的数据集。这个数据集的指标 T 作为预测指标，一系列从变量选择过程得到的变量将作为预测变量。在3.3.1节中已经看到，我们的真正目标是预测在任何时间 t 的正确交易信号。给定我们在前一节生成的数据，如何进行这种预测呢？下面我们来探索获取这种正确交易信号预测值的两种途径。

第一种预测任务是应用 T 值作为目标变量，然后尝试获取模型，该模型应用预测变量的信息预测 T 值。这和第2章考虑的多元线性回归任务相似。如果应用这种方法，我们需要把模型的预测值转换成交易信号。这意味着需要给出界限值来决定预测的 T 值对应三种交易信号的哪一种，转换公式如下：

$$signal = \begin{cases} \text{卖出} & T < -0.1 \\ \text{持有} & -0.1 \leqslant T \leqslant 0.1 \\ \text{买入} & T > 0.1 \end{cases} \tag{3-6}$$

这里选择界限值0.1和-0.1纯粹是启发式的，也可以使用其他的界限值。然而，这两个界限值的意义是，生成 T 值的10日间，至少有四天的平均日价高于目前收盘价2.5%（$4*0.025=0.1$）。如果决定使用其他的界限值，需要考虑的是，太大的界限绝对值将导致更少的信号，而太小的界限值可能会导致在太小的市场变化时进行交易，从而招致更大的风险。本书R包中的函数 trading. signals( )，可以进行上面的转换，它把数值型的 T 值转变为三个可能的值："s"、"h" 和 "b"，分别代表卖出、持有和买入行动。

第二个可供选择的预测任务是直接预测交易信号，这意味着把第 d 天的正确信号作为目标变量。怎样才能获得这些正确信号呢？这里仍然使用指标 T 和式（3-6）。对于可用的历史数据，我们通过接下的10天来计算指标 T 的值，然后用式（3-6）中的界限值来确定信号，从而得到每117一天的交易信号。在第二个预测任务中，目标变量是一个名义变量。目标变量为名义变量的预测问题称为分类任务⊖。分类任务（问题）和回归任务的主要区别是目标变量的类型。回归任务有一个数值型目标变量（例如，我们案例中的 T 指标），而分类任务的目标变量是取值为有限个值的名义变量。不同的方法和技巧可以用于上面的两类问题。

R的 xts 添加包中的函数基于数值型数据。xts 包中对象的数据属性必须是向量或者矩阵，即它们是单一模式的数据。所以它不允许训练集数据的一列为名义变量（R的因子），而其他列为数值型变量。为了克服 xts 的这个问题，我们在 xts 外进行建模的整个过程。下面你会看到，这样比较简单且没有任何像 xts 那样的限制。我们用 xts 提供的功能来进行数据子集的选取和绘图，

⊖ 有些统计机构称该方法为"判别任务"。

建模阶段可以不必应用 xts 的功能。

下面的代码构造本节下面部分两个预测任务的预测模型所应用的数据结构。

```
> Tdata.train <- as.data.frame(modelData(data.model,
+                         data.window=c('1970-01-02','1999-12-31')))
> Tdata.eval <- na.omit(as.data.frame(modelData(data.model,
+                         data.window=c('2000-01-01','2009-09-15'))))
> Tform <- as.formula('T.ind.GSPC ~ .')
```

上面代码中的 Tdata. train 和 Tdata. eval 是两个数据框，它们的数据分别用于模型训练阶段和模型评价阶段。我们使用数据框作为基本数据结构，它允许分类任务所要求的混合模式数据集。我们用函数 trading. signals( ) 产生数值型目标变量的相应信号，然后用信号来替代原来的目标变量。在模型选择和比较过程中没有应用数据框 Tdata. eval，在评估最后得到的最优模型时才应用该数据框。调用函数 na. omit( ) 是必要的，它可以避免在数据框结束部分由于没有未来数据计算 T 值所导致的 NA 值。

3.3.4 模型评价准则

3.3.3 节中描述的预测任务获得的模型可以给出未来市场方向的预测值。对回归任务而言，这个预测值是一个数值（即 T 值的预测值）；对分类任务而言，这个预测值是一个信号。上面已经看到，即使在回归任务的情况下，我们用一个界限值将回归模型的预测值转换为信号。在 3.5 节中，我们将描述几个根据这些预测信号进行市场行为的交易策略。本节不讨论如何评价模型信号预测问题。我们不考虑 T 指标的数值预测评价。由于我们不直接使用预测的数值指标 T，因此评价数值型预测值与我们的目标不相干。由于我们只对交易信号感兴趣，所以甚至有人可能质疑这些回归任务是否有意义。这里我们仍然保持这些数值型的预测任务，因为不同的交易策略可能会利用数值型预测值的优势，比如，当开盘时，可以根据数值型预测值的大小来决定投资金额。决定开盘时的 T 值，如果远远大于界限值（$T>0.1$ 决定买入和 $T<-0.1$ 决定卖出），可以导致更大的投资。 118

可以通过测量错误率来衡量信号预测，错误率的定义为

$$\text{error. rate} = \frac{1}{N}\sum_{i=1}^{N} L_{0/1}(y_i, \hat{y}_i) \tag{3-7}$$

这里 $\hat{y}_i$ 是第 i 个测试数据的模型预测值，而真实的类标签为 y_i，$L_{0/1}$ 是 0/1 损失函数，其定义为：

$$L_{0/1}(y_i, \hat{y}_i) = \begin{cases} 1 & \hat{y}_i \neq y_i \\ 0 & \hat{y}_i = y_i \end{cases} \tag{3-8}$$

有时候经常采用上面损失函数的对立测度，即精确度，定义为 1 - error. rate。

式（3-7）和式（3-8）中定义的统计量都是用来把模型的预测值和未来 k 天市场上真实发生的结果进行比较。

精确度（或者错误率）被证明不是衡量分类问题好坏的最佳指标。事实上，在三个可能的结果中，它们极不平衡。在金融市场中，大的价格波动比较少，所以“持有”这一结果出现的次数大大超过另外两个结果出现的次数[⊖]。这就意味着精确度值将主要由模型在出现最多的结果（持有）上的性能来决定。然而，“持有”却不是我们交易所关心的结果。我们需要在“罕见”事件（买入或者卖出）上预测准确的模型。买入或者卖出事件才是导致市场行为并有潜在利润的事件，也是我们应用的最终目标。

⊖ 很明显这取决于你所建立的目标边际利润。但是，为了能覆盖交易成本，应该设定该边际利润足够大，所以，这里的“罕见”的确是事实情况。

金融市场预测是罕见事件驱动应用的一个例子。基于事件的预测任务通常由决策精确度指标和回溯精确度指标来衡量，这两个指标可以集中于所关注事件的评估而不是常见事件的评估（我们案例中的“持有”信号即为常见事件）。决策精确度衡量模型给出的事件信号的正确百分比；而回溯精确度则是指模型给出的事件信号占事实存在的百分比。通过分类矩阵（又称为混淆矩阵）可以方便地计算这两个指标。分类矩阵是通过比较模型的预测值和真实值，然后汇总得到的。表3-1给出了本案例的分类矩阵。

119

通过表3-1，本案例的决策精确度和回溯精确度指标正式定义如下：

$$\text{Prec} = \frac{n_{s,s} + n_{b,b}}{N_{.,s} + N_{.,b}} \tag{3-9}$$

$$\text{Rec} = \frac{n_{s,s} + n_{b,b}}{N_{s,.} + N_{b,.}} \tag{3-10}$$

表3-1 预测交易信号的分类矩阵

		预测结果			
		买入	持有	卖出	
真实结果	卖出	$n_{s,s}$	$n_{s,h}$	$n_{s,b}$	$N_{s,.}$
	持有	$n_{h,s}$	$n_{h,h}$	$n_{h,b}$	$N_{h,.}$
	买入	$n_{b,s}$	$n_{b,h}$	$n_{b,b}$	$N_{b,.}$
	买入	$n_{.,s}$	$n_{.,h}$	$n_{.,b}$	N

也可以单独对某个特定的信号（例如买信号或者卖信号）计算其独立的决策精确度和回溯精确度，例如：

$$\text{Prec}_b = \frac{n_{b,b}}{N_{.,b}} \tag{3-11}$$

$$\text{Rec}_b = \frac{n_{b,b}}{N_{b,.}} \tag{3-12}$$

经常把决策精确度和回溯精确度合并为一个统计量，称为F度量（Rijsbergen，1979），合并后的统计量为：

$$F = \frac{(\beta^2 + 1) \cdot \text{Prec} \cdot \text{Rec}}{\beta^2 \cdot \text{Prec} + \text{Rec}} \tag{3-13}$$

这里$0 \leqslant \beta \leqslant 1$，它控制回溯精确度相对决策精确度的相对重要性。

3.4 预测模型

在本节中，我们将探讨一些模型，通过运用这些模型来完成3.3节所确定的预测任务。我们主要通过研究模型处理非线性回归问题的优劣来选择最佳模型，这就是我们案例中所要研究的问题。当然，许多其他的方法也可以用来解决这个问题，在研究股票收益率这样一个领域中，任何一个更好的研究方法都必然需要经历更多的比较和更多的选择。在本书上下文的背景下，这样的探索将由于其需要计算空间和计算能力方面的费用而显得没有意义。

120

3.4.1 如何应用训练集数据来建模

复杂的时间序列问题通常有着不同的表现形式，比如序列由波动大的阶段到相对较平稳的阶段，或者是有某种趋势倾向的序列。这些类型的序列通常称为是非平稳时间序列，在某些模型的基本假设条件下，非平稳序列将会带来严重的问题。这是显而易见的，比如运用我们案例中的数据，画出价格的时间序列图。为了克服由非平稳序列带来的消极影响，我们可

以使用一些方法来尝试着消除这些影响，比如使用适用于原始时间序列的一些转换方法，用收益百分比的变化来替换原来的绝对价格就是一种转换方法，也可以对已经获得的数据有选择地使用。假设需要用给定期间的训练集数据来建立一个预测模型，并使用该模型获得其测试时间段的预测值。标准的做法是用训练集建立模型，然后将模型应用到测试集获取预测值。如果我们有理由相信序列在某个时间点发生了变化，那么用同一个模型来预测测试时间段的数据将不是最好方法，尤其是当测试时间段存在一个改变点时，它将会严重破坏原有模型的性能。在这种情况下，我们需要改变或者调整模型，使模型适应最近的数据，这样模型就能获取当前数据的变化。

在时间序列问题中，测试案例中有一种隐含的时间顺序。在上下文中，假设当我们获得一个时间序列在 i 时的预测时，所有时间标记小于 i 的时期 k（$k<i$）的数据就已经属于过去。这就意味着，假设我们已经知道这些过去时期的目标变量的真实值，那么我们就可以放心地使用这些信息。因此，如果在测试集时间序列的某个时间点 m，我们有信心相信时间序列发生了变化，那么应该把测试集时间序列时间点 m 之前的所有观测值包含到初始训练集数据中，然后用这个包含新的变化信息的训练集数据来更新预测模型，这样就可以提高预测模型对未来情况的预测性能。更新模型的一种方式就是使用新的训练案例数据来改变原来的模型。这些方法通常称为增量学习，因为它更新模型以适应最新的信息而无须从头开始。目前还没有太多的建模技巧应用这种更新模型的方式，尤其是在 R 中，没有很多的模型可以采用这种方式。在本书中，我们将按照其他的方法来更新模型，使用新的训练数据集来重新建立新的模型。这显然在计算方面更有难度，尤其是数据到达速度很快，并且要求几乎实时地给出模型和决策时，这种方法就不适用了。这个问题在应用研究中很常见，即数据流问题。在我们的应用程序中，我们在每天收市后做出决策，因此速度不是一个关键问题[⊖]。假设我们使用一种重新学习的方法，我们有两种基本的形式将新的案例纳入训练集中。不断增加窗口的方法只是简单地将它们添加到当前的训练集，从而不断地扩大样本集。这种方法的最终问题在于，由于我们假设最近的数据是有助于建立更好的模型，所以我们也会考虑我们最早的那部分训练数据是不是已经过时了，是不是可能会降低模型的精确性呢？基于这些考虑，在滑动窗口方式下，在删除训练集中最原始数据的同时，加入最新的观察数据，从而保持训练集的大小不变。

121

扩大和滑动窗口的方法都涉及一个关键的决定：什么时候应该通过纳入更新的数据来改善或者调整模型？解决这个问题主要有两种方法。第一种方法需要通过检查来估计模型的预测能力从什么时候开始降低。如果我们观察到在某个时候，模型的预测性能突然降低，我们就可以认为在这时模型发生了改变。这种方法最主要的挑战就在于给出模型性能变化的合理估计。我们需要尽快地检测到模型性能的变化，但又不能对模型没有捕获的一些噪声案例过度反应。第二种更简单的方法是在常规时间的基础上更新模型，也就是说，每隔 w 个测试用例，用更新的数据来建立新的模型。在本案例中，我们就采用这种方法。

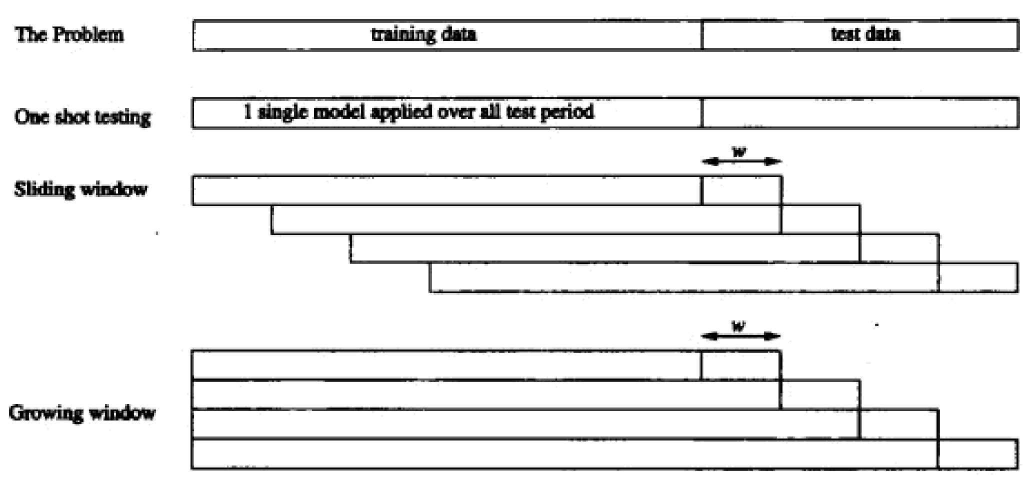
总之，我们考虑每一个模型应用会用到以下三种不同的方法：1）所有的测试时段都使用一个模型；2）每隔 w 天更新数据增长窗口；3）用每隔 w 天的数据滑动窗口。图 3-3 说明了这三种方法。

趋势变化的参考文献

时间序列数据的趋势变化问题是统计过程控制（例如，Oakland，2007）领域长时间研究的课题，它使用控制图检测数据的突变点。这一课题受到数据流的影响（例如，Gama and Gaber，2007），人们在数据挖掘领域的兴趣不断增加。多部作品（Gama et al.，2004；Kifer et al.，

⊖ 如果是实时交易，即在交易日中间交易，则速度就是需要关心的一个问题。

2004；Klinkenberg，2004）已经解决了问题：如何检测制度的变化，以及如何学习和运用在这些变化方面的模型。



122

图 3-3　三种形式获得测试时段数据的预测

3.4.2　建模工具

在本节中，我们简要介绍建模的方法，将使用这些方法来完成我们的预测任务，并说明在 R 中如何使用这些方法。

3.4.2.1　人工神经网络

人工神经网络（Artificial Neural Network，ANN）经常在金融预测中使用（例如，Deboeck，1994），因为运用人工神经网络可以处理高度非线性问题。R 的添加包 nnet 可以实现前馈神经网络。这种类型的神经网络是最常用的，也是我们将要使用的神经网络。

人工神经网络中由相互联系的计算单元（即神经元）构成。每个神经元执行两次连续的计算：输入的线性组合；之后对前面结果的非线性计算得到的输出值作为神经网络的下一个神经元的输入。每个神经元连接都有一个相关联的权重。要构建人工神经网络，先要建立网络体系结构，然后使用一种算法来计算出神经元之间的连接权重。

前馈人工神经网络按层来组织神经元。第一层包含网络输入神经元，训练集的观测值通过这些输入神经元传递给网络。最后一层包含了任何情况下传递给神经网络输入神经元的神经网络预测值。在这两层之间，通常有一个或多个“隐藏”层神经元。权重更新算法，比如反向传播法，试图获得能够优化某个误差标准的连接权重，也就是试图确保网络输出与提交给神经网络模型的训练集个案一致。这是通过在网络输入结点多次传入训练个案来进行迭代的一个过程，在网络输出结点获得预测值并计算出各自的预测误差后，通过更新网络中的权重来减小模型的预测误差。这种迭代过程反复进行，直到满足一定的收敛准则。

使用 R 中的添加包 nnet（Venables and Ripley，2002）中的一个函数实现了带有隐藏层的前馈神经网络。通过该函数获得的网络，既可以用于分类问题，也可以用于回归问题，因此它适用于我们的预测任务（见 3.3.3 节）。

123

人工神经网络对预测问题中变量的尺度敏感。这种情况下，在将数据应用到神经网络前，先进行数据转换是很有意义的。这样就可以避免神经网络模型的性能受到变量尺度的影响。在我们的案例中，先进行数据的标准化处理，使所有变量均具有零均值和标准差为 1。通过下面的公式，可以很容易地对数据集的每一列应用下列公式进行转换：

$$y_i = \frac{x_i - \bar{x}}{\sigma_x} \tag{3-14}$$

其中，$\bar{x}$ 是原始变量 X 的均值，σ_x 是变量 X 的标准偏差。

函数 scale( ) 可以用来进行上面的数据变换，在本书的 R 添加包中，还可以找到函数

unscale( )，这个函数可以进行标准化过程的逆运算，将转换后的数据转变为原来尺度的数据。下面是一个非常简单的例子，用来说明在 R 中如何获取和使用这种类型的神经网络：

```
> set.seed(1234)
> library(nnet)
> norm.data <- scale(Tdata.train)
> nn <- nnet(Tform, norm.data[1:1000, ], size = 10, decay = 0.01,
+      maxit = 1000, linout = T, trace = F)
> norm.preds <- predict(nn, norm.data[1001:2000, ])
> preds <- unscale(norm.preds, norm.data)
```

在默认情况下，函数 nnet( ) 以在区间 [-0.5, 0.5] 的随机值来设置结点之间链接的初始权重，这意味着连续运行两次有完全相同参数的同一函数，得到的实际结果可能不同。为了确保可以得到相同的结果，我们增加了调用函数 set. seed( )，这个函数可以初始化随机数发生器中的种子值，这样就确保了可以得到与本书中一样的结果。在这个示例中，我们用开始的 1000 个训练集观测值来创建神经网络，并用接下来的 1000 观测值来测试这个预测模型。将训练数据标准化后，我们调用函数 nnet( ) 来建立模型。前两个参数是 R 中任何模型的函数中都有的参数。模型的函数形式通过一个公式来具体化，训练集数据用于建立模型。我们也使用 nnet( ) 函数中的一些参数。也就是说，参数 size 用来指定隐藏层中的结点个数。这里使用的参数 size 的值没有什么神奇配方，人们通常会尝试几个值来观察神经网络的行为。尽管如此，可以合理地假设该参数的值小于问题中预测变量的个数。参数 decay 控制反向传播算法权重的更新率。而且，实验和查错是最重要的方法。最后，参数 maxit 控制权重收敛过程所允许使用的最大迭代次数；而参数 linout = T 告诉该函数是处理回归问题；参数 trace = F 是用来避免一些和优化过程有关的结果被输出。

124

函数 predict( ) 可以用来获得测试数据集的神经网络预测值。取得这些预测值后，我们使用本书 R 添加包中提供的函数 unsacale( ) 转换为原来尺度的数据。这个函数的第一个参数是预测值，第二个参数是含有标准化数据的对象。后一个参数中的对象是必要的，因为该对象中存储了用于标准化数据的均值和标准差[⊖]，这两者是逆转标准化过程所必需的。

让我们来评估人工神经网络模型预测测试集信号的准确性。我们可以将数值型预测值转换成信号，然后使用 3.3.4 节中的统计方法。

```
> sigs.nn <- trading.signals(preds, 0.1, -0.1)
> true.sigs <- trading.signals(Tdata.train[1001:2000, "T.ind.GSPC"],
+      0.1, -0.1)
> sigs.PR(sigs.nn, true.sigs)

    precision    recall
s   0.2101911 0.1885714
b   0.2919255 0.5911950
s+b 0.2651357 0.3802395
```

分别给定买入和卖出决策的界限值，函数 trading. signals( ) 可以将预测数值转换成信号。函数 sigs. PR( ) 可以获得我们关心的两类事件以及决策精确度和回溯精确度矩阵。这些值表明人工神经网络的预测性能并不是很好。实际上，会得到相当低的预测精确度，回溯精确度值也不是很好。但是，后者比较差所导致的问题不是太严重，因为它基本上意味着失去机会而不意味着有成本损失。另一方面，预测精确度的较小值意味着该模型频繁给出错误信号。如果这些信号被用来交易，就可能导致严重的损失。

⊖ 作为对象的属性。

人工神经网络也可以用于分类问题。对于这类问题，在网络拓扑方面最主要的区别在于不是一个单一的输出单位，我们将有和目标变量值（有时称为类变量）一样多的输出单位，这些输出单位的每一个都将产生各自类值的概率估计。这意味着，对于每一个测试个案，人工神经网络可以产生一组概率值，每一个概率值相应于一个可能的类值。

使用 nnet( ) 函数来完成这个任务与使用该函数解决回归问题很相似。使用训练数据并应用125下面的代码进行演示：

```
> set.seed(1234)
> library(nnet)
> signals <- trading.signals(Tdata.train[, "T.ind.GSPC"], 0.1,
+     -0.1)

> norm.data <- data.frame(signals = signals, scale(Tdata.train[,
+     -1]))
> nn <- nnet(signals ~ ., norm.data[1:1000, ], size = 10, decay = 0.01,
+     maxit = 1000, trace = F)
> preds <- predict(nn, norm.data[1001:2000, ], type = "class")
```

这里的参数 type = "class" 是用于获得测试集个案的类标签，而不是概率的估计值。在神经网络预测中，可以计算出模型的预测精确度和回溯精确度，代码如下：

```
> sigs.PR(preds, norm.data[1001:2000, 1])

    precision    recall
s   0.2838710 0.2514286
b   0.3333333 0.2264151
s+b 0.2775665 0.2185629
```

上面结果中，预测精确度和回溯精确度的值还较低，但是都高于回归任务中相应的值。

基于神经网络的参考文献

Rojas（1996）的书是基于神经网络的一本不错的参考书。对于更多金融方面的读物，Zirilli（1997）的书是一本很好的和容易阅读的书。题为“Artificial Neural Networks Forecasting Times Series”（Rogers and Vemuri，1994）的论文集是另外一个好的引用源和参考文献。Deboeck（1994）书的第一部分专门提供了神经网络应用程序交易的几个章节。McCulloch 和 Pitts（1943）提出了第一个人工神经元模型，这项工作是由 Ronsenblatt（1958）、Minsky 和 Papert（1969）推广的。反向传播法，最常用的权重更新方法，虽然经常归功于 Rumelhart 等（1986），但根据 Rojas（1996）的书，这个方法是由 Werbos（1974，1996）发明的。

3.4.2.2　支持向量机

支持向量机（Support Vector Machine，SVM）[⊖]和人工神经网络一样，也是一种建模工具，可以用于回归和分类问题。基于其成功应用到多个领域和其强大的理论背景，支持向量机已经受到越来越多不同研究领域的关注。Vapnik（1995，1998）、Shawe-Taylor 和 Cristianini（2000）是支持向量机的两个重要参考文献。Smola 和 Scholkopf（2004，1998）出版了一本极好的支持向量机指南，概述了支持向量机用于回归的基本思想。R 中有多个添加包实现了支持向量机，在这些126添加包中，我们可以参考由 Karatzoglou 等（2004）提供的带有多种功能的添加包 kernlab，我们也可以使用由 Dimitriadou 等（2009）提供的带有 svm( ) 函数的添加包 e1071。

支持向量机的基本思想是，将原始数据映射到一个新的高维空间中，在这个新的高维空间中，有可能应用线性模型来获得一个超平面来进行分离，例如在分类任务中，分离问题中的不同

⊖ 可以在网站：http://www.kernel-machines.org 上得到该类模型的大量信息。

类别。将原始数据映射到这一新的空间是在所谓的核函数的帮助下进行的。支持向量机是作用在由核函数所引入的对称表示的线性机。

在新的对称表示下进行超平面分割，这是通过最大化不同类别之间个案的分割边际来进行的。参见图 3-4。这是一个优化问题，经常用二次规划来解决。软边界方式允许将比例很小的个案划分到“错误”的类别，这些方式导致一定的“损失”。

在支持向量回归中，这个过程很相似，主要区别在于误差和相关损失的计算。这通常借助于所谓的 ε 不敏感损失函数 $|\xi|_\varepsilon$，该函数形式如下：

$$|\xi|_\varepsilon = \begin{cases} 0 & |\xi| \leqslant \varepsilon \\ |\xi| - \varepsilon \end{cases} \tag{3-15}$$

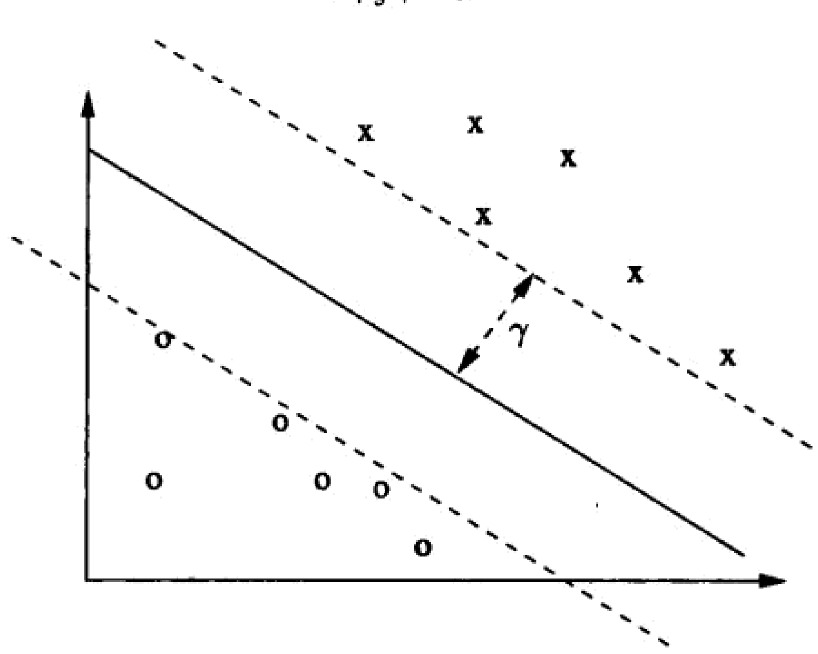


图 3-4 支持向量机边际最大化

下面，我们将提供使用 R 中这类模型的简单例子。我们从使用添加包 e1071 中的函数进行回归任务开始，代码如下：

```
> library(e1071)
> sv <- svm(Tform, Tdata.train[1:1000, ], gamma = 0.001, cost = 100)
> s.preds <- predict(sv, Tdata.train[1001:2000, ])
> sigs.svm <- trading.signals(s.preds, 0.1, -0.1)
> true.sigs <- trading.signals(Tdata.train[1001:2000, "T.ind.GSPC"],
+     0.1, -0.1)
> sigs.PR(sigs.svm, true.sigs)

    precision     recall
s   0.4285714 0.03428571
b   0.3333333 0.01257862
s+b 0.4000000 0.02395210
```

在这个例子中，我们使用了函数 svm()，除参数 gamma 和 cost 外，大部分的参数我们都采用默认值。在本节中，该函数使用了一个径向基核函数：

$$K(x,y) = \exp(-\gamma \times \|x-y\|^2) \tag{3-16}$$

其中 γ 是一个用户参数，在上面的函数调用中，它的值设置为 0.001。（在函数 svm() 中，该参数的默认值为 1/ncol(data)）。

参数 cost 给出违反边际所引入的损失。你可以参考函数 svm 的帮助页面以获取该参数和其他参数的细节。

我们可以观察到，支持向量机模型的决策精确度值比人工神经网络好许多，尽管回溯精确

度的值很低。

下一步，我们考虑分类任务，这次使用 R 的 kernlab 添加包，代码如下：

```
> library(kernlab)
> data <- cbind(signals = signals, Tdata.train[, -1])
> ksv <- ksvm(signals ~ ., data[1:1000, ], C = 10)

Using automatic sigma estimation (sigest) for RBF or laplace kernel

> ks.preds <- predict(ksv, data[1001:2000, ])
> sigs.PR(ks.preds, data[1001:2000, 1])

    precision    recall
s   0.1935484 0.2742857
b   0.2688172 0.1572327
s+b 0.2140762 0.2185629
```

我们使用 kernlab 添加包中的函数 ksvm( )，其中的参数 C 用来指定违反约束的不同损失，该参数的默认值为 1。除此之外，其他参数使用默认值，例如，在分类时用的默认参数是径向基核函数。可以通过函数 ksvm( ) 的帮助页面获取更多的细节。

127 ~ 128

支持向量机分类的结果并不如支持向量机回归的结果好。这并不意味着我们声明这是用支持向量机技术所能获得的最好结果。这些都只是说明如何在 R 中使用这些建模技术的简单例子。

3.4.2.3 多元自适应回归样条

多元自适应回归样条（MARS）（Friedman，1991）是自适应回归模型（Hastie and Tibshirani，1990）的一个例子。一个多元自适应回归样条模型具有以下一般形式：

$$\text{mars}(\boldsymbol{x}) = c_0 + \sum_{i=1}^{k} c_i B_i(\boldsymbol{x}) \tag{3-17}$$

其中 c_i 是常数，$B_i(\boldsymbol{x})$ 是基函数。

基函数可以有多种不同的表现形式，从简单常量到描述两个或多个变量相互关系的函数。但是，最常见的基函数是所谓的铰链函数，有如下形式：

$$H[-(x_i - t)] = \max(0, t - x_i) \quad H[+(x_i - t)] = \max(0, x_i - t)$$

其中 x_i 是一个预测变量，t 是该预测变量的界限值。图 3-5 给出这两个函数的一个例子。

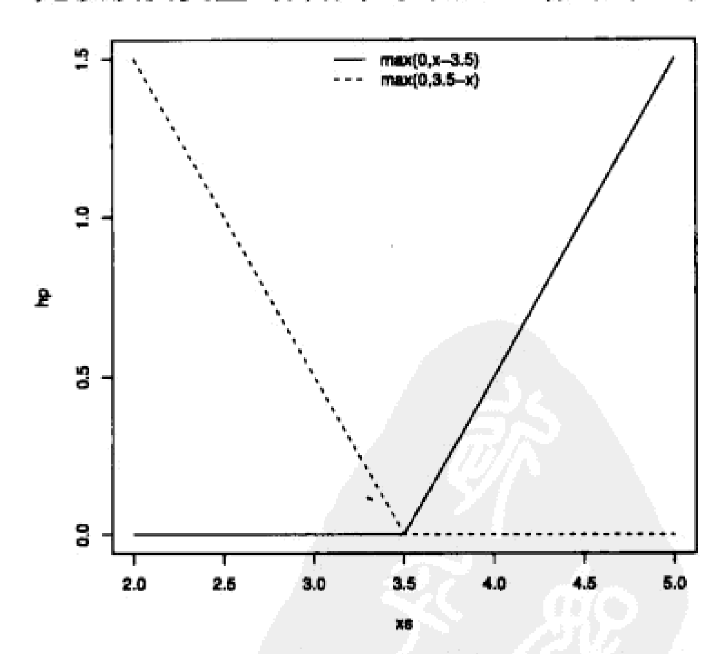


图 3-5　两个具有相同阈值的铰链函数的例子

在 R 中已经至少有两个添加包实现了多元自适应回归样条模型。添加包 mda（Leisch et al.，2009）包含了函数 mars( )，该函数实现了多元自适应回归样条方法。添加包 earth（Milborrow，

2009）中的函数 earth( ) 同样也实现了多元自适应回归样条方法。从建模函数提供的基于公式的接口来看，函数 earth( ) 有着能够遵循更标准的 R 构架的优势。同时添加包 earth 还实现了其他添加包所不具有的功能，因此我们选择添加包 earth。 129

下面是应用函数 earth( ) 进行回归的代码：

```
> library(earth)
> e <- earth(Tform, Tdata.train[1:1000, ])
> e.preds <- predict(e, Tdata.train[1001:2000, ])
> sigs.e <- trading.signals(e.preds, 0.1, -0.1)
> true.sigs <- trading.signals(Tdata.train[1001:2000, "T.ind.GSPC"],
+     0.1, -0.1)
> sigs.PR(sigs.e, true.sigs)

    precision    recall
s   0.2785714 0.2228571
b   0.4029851 0.1698113
s+b 0.3188406 0.1976048
```

得到的结果与支持向量机的回归结果相差不大，决策精确度约为 30%，回溯精确度要更低一些。

多元自适应回归样条只适用于回归问题，所以在分类问题方面我们就不再举例说明。

关于多元自适应回归样条的参考文献

多元自适应回归样条可以参考具有权威性的 Friedman（1991）的原始文章。这是一篇很好的文章，它提供了所有关于推动多元自适应回归样条的发展以及系统中技术应用的所有细节。这篇文章还包括了其他科学家工作意见和有趣的讨论。

3.5 从预测到实践

这节描述如何应用上节的模型所得到的预测信号。给定模型输出的一组信号，可有许多方式将它们运用于市场的交易决策。

3.5.1 如何应用预测模型

在本案例中，我们假设在期货市场进行交易。期货市场是在合约基础上进行交易的，合约规定在未来某个确定的时间、以未来市场决定的价格买入或者卖出商品。这些合约的技术细节超过了本书的范围。但是，从客观的角度看，这意味着我们的交易系统将可以采取两种交易头寸： 130 多头头寸（也称为多头仓位）和空头头寸（也称为空头仓位）。多头头寸是指在 t 时刻、以价格 p 买入商品，随后在时刻 $t+x$ 卖出。当交易者预期未来价格上涨时，这样的头寸对交易者来说是有意义的，进行这种交易使他获取利润。做空头头寸时，交易者在时刻 t、以价格 p 卖出证券，同时他们有义务在未来买回同样的证券。由于可以借入证券的交易模式，这种空头头寸也是可能的（有关该模式的细节，请参阅其他文档，例如维基百科）。因为当证券价格下降时，他们可以在时刻 t 后的某个时刻买入并偿还所借证券，从而获利。简略地说，当认为价格会下降时开空头仓位，认为价格会上涨时开多头仓位。

给出一组交易信号，可以有许多方式在期货市场上应用它们。下面将描述我们的模型实验中将应用并进行比较的一些具有可信性的交易策略。由于空间和时间限制，这里不可能进一步探讨该重要问题。但是，读者可以练习一些其他具有可信性的策略，并可以开发和尝试其他可能的策略。

我们应用的第一个交易策略机制是这样的。首先，将在一天股票收盘时执行所有的决定，也就是在了解当前所有的日报价信息后。假设在某日 t 收盘时，我们的模型提供证据表明价格正在

下跌，即预测出一个很低的 T 值或者一个卖出的信号。如果我们现在持有多头头寸，那么模型给出的信号将被忽视；如果我们现在没有持有多头仓位，我们就可以发出卖空指令，建立空头头寸。当这个指令在未来某个时候以价格 pr 执行时，我们将立即跟上其他两个指令。第一个指令是一个限价购买的指令，限制价格为 pr $- p\%$，这里 $p\%$ 为目标收益率。这类指令只有当市场价格达到或低于限制价格时才会执行。该指令给出了当前卖空操作的利润目标。我们将会等待 10 天以达到这个目标。如果指令在最后期限前没有执行，我们将以第 10 天的收盘价买入。第二种指令是止损指令，价格上限是 pr $+ l\%$。这种指令的目的是限制我们上面做卖空操作的最终损失。如果市场价格到达限定价格的 pr $+ l\%$，那么该指令将执行，因此我们的损失可以限制在 $l\%$。

如果模型提供的预测表明：价格将在近期上涨，表示指标 T 的预测值较高或者给出买入信号时，我们将考虑开多头仓位。只有当我们现在没有任何仓位时，才会建这类仓位。带着这个目标，我们会在时刻 t、以价格 p 完成一个买入指令。和前面一样，我们将立即跟上两个新指令。第一个指令将是卖出限制价指令，目标价格为 pr $+ p\%$，只有当价格高于或等于 pr $+ p\%$ 时，这个指令才执行。这种限价指令和先前一样有 10 天的期限。第二种指令是卖出止损限制，限制价格为 pr $- l\%$，这将再次限制我们的最终损失为 $l\%$。

131

第一种策略有些保守，因为它在任一时刻只能有一个仓位。此外，在经过 10 天等待目标利润后，立即平仓。我们也会考虑一个有更多“风险”的交易策略。后一策略与前一个类似，如果有预测信号表明价格上升，如果有足够的资金，那么总是开新的多头仓位。另外，我们可以一直等待直到所持仓位到达目标利润或到达最大允许的损失。

我们只考虑这两个主要的交易策略，这两个策略所应用的参数可以有细微的变化（例如，持有期、预期收益率，或在每个仓位上投资的资金量）。如前所述，这里所选择的两个策略主要以说明为目的。

3.5.2 与交易相关的评价准则

3.3.4 节阐述过的模型性能的衡量标准不能直接应用到本案例的应用中。本案例模型性能的衡量需要与经济效益相结合。在本案例的背景下，像经济效益和一些金融工具所暴露的风险等指标是本案例的模型需要考虑的关键指标。这些指标可能就需要一章的篇幅来进行说明。R 的性能分析添加包 PerformanceAnalytics（Carl and Peterson，2009）实现了分析某些交易算法性能的诸多金融指标，例如分析本章交易的一些金融指标。我们将使用这个添加包所提供的函数收集我们需要的经济效益指标信息。我们的交易评估将关注方法的整体效果、风险暴露和根据模型提示所建立的每个仓位（头寸）的平均结果。3.7 节介绍的对我们给出的交易系统的最终评估中，我们将使用这个添加包所提供的工具进行更深入的交易系统性能分析。

我们将使用以下 3 个指标来衡量交易的整体结果：1）初始资本与测试期期末资本之间的净差额（有时称为利润/损失）；2）净差额所代表的百分比收益率；3）买入并持有策略的超额回报。这个策略包括在测试期开始开多头仓位和等待到最后平仓。用购买并持有的收益来衡量我们的交易策略和这个简单策略之间的差异。

对于与风险相关的测量，我们将使用夏普比率系数测量每单位风险的回报，风险由收益的标准偏差来衡量。我们也将计算出跌幅最大值（最大回撤），以测量出模型的最大连续累积损失。对于交易者，这是一个重要的风险测量，若系统有一个严重的最大回撤，就可能导致没有资金来投入该交易系统，因为投资者肯定会害怕这些连续亏损并撤出他们的资金。

132

最后，根据测试期所持有仓位的数量、每个仓位的平均收益、盈利仓位的百分比以及其他相关性不大的绩效指标来评估它们的效果。

3.5.3 模型集成：仿真交易

本节将描述如何实现前面几节通过模型给出的信号来进行交易的想法。本书给出的添加包

提供的函数 trading. simulator( ) 把上面的想法结合在一起，实现对任何模型给出的信号进行交易仿真的功能。这个函数的主要参数是仿真期间的市场报价和同一时期的模型信号。另外两个参数是自定义交易策略的函数及其参数列表。最后，我们也可以指定每一笔交易的成本和交易者能够提供的初始资本。模拟器会在每天收盘时调用用户提供的交易策略函数，而用户交易策略函数应该返回它需要模拟器执行的交易指令。模拟器在市场上执行这些指令并用多个数据结构记录下所有的交易活动。模拟的结果是一个 tradeRecord 类对象，它包含该次仿真的信息。这个对象可以用于获取经济评估指标的函数或绘制交易活动图表的函数，这些将在之后的操作中看到。

在给出这种类型的仿真实例之前，我们需要提供用于模拟器的交易策略函数的更多细节。应该使用某种协议来编写这些函数，也就是说，它们应该意识到模拟器将如何调用它们，同时它们如何返回模拟器所期望的信息。

每一个股票交易日 d 收盘之后，模拟器用四个主要参数以及用户提供的任何其他参数来调用交易策略函数。这四个参数是：1）含有直到 d 天的预测信号的一个向量；2）市场报价（直到 d 天）；3）当前持仓情况；4）交易者当前可使用的资金。当前持仓情况是一个矩阵，该矩阵的行数与 d 天收盘时的持仓数相等。该矩阵有 4 列：“pos. type”为 1 代表多头仓位，－1 代表空头仓位；“N. stocks”是该仓位中的股票数；“Odate”是指开仓的日期（日期在 1 ~ d 之间）；“Oprice”是开仓时的价格。这个矩阵的行名称含有仓位的标识符（ID），当我们需要模拟器平仓某个特定仓位时需要用到该 ID。

所有这些模拟器提供的信息能确保用户可以定义一个较大的交易策略函数集合。用户自定义的策略函数应该返回一个含有交易指令集合的数据框，返回的交易指令将由模拟器执行。这个交易指令数据框应包括以下信息（列）：“order”为 1 代表买入指令，－1 代表卖出指令；“order. type”为 1 代表需要立刻执行的市场指令（实际上是以次日的开盘价执行），2 代表限价指令，3 代表止损指令；“val”是指开仓指令中的交易股票数量；NA 是指平仓指令，或者限价指令与止损指令中的目标价格；“action”的值是“open”代表开新仓位指令，或者取值为“close”代表平仓已有仓位的指令；最后，“posID”如果取值非空，其内容是需要平仓的仓位 ID。 133

下面是用户定义的交易策略的示例：

```
> policy.1 <- function(signals,market,opened.pos,money,
+                      bet=0.2,hold.time=10,
+                      exp.prof=0.025, max.loss= 0.05
+                      )
+   {
+     d <- NROW(market) # this is the ID of today
+     orders <- NULL
+     nOs <- NROW(opened.pos)
+     # nothing to do!
+     if (!nOs && signals[d] == 'h') return(orders)
+
+     # First lets check if we can open new positions
+     # i) long positions
+     if (signals[d] == 'b' && !nOs) {
+       quant <- round(bet*money/market[d,'Close'],0)
+       if (quant > 0)
+         orders <- rbind(orders,
+               data.frame(order=c(1,-1,-1),order.type=c(1,2,3),
+                     val = c(quant,
+                             market[d,'Close']*(1+exp.prof),
+                             market[d,'Close']*(1-max.loss)
+                            ),
```

```
+                            action = c('open','close','close'),
+                            posID = c(NA,NA,NA)
+                           )
+                         )
+
+     # ii) short positions
+     } else if (signals[d] == 's' && !nOs) {
+       # this is the nr of stocks we already need to buy
+       # because of currently opened short positions
+       need2buy <- sum(opened.pos[opened.pos[,'pos.type']==-1,
+                                  "N.stocks"])*market[d,'Close']
+       quant <- round(bet*(money-need2buy)/market[d,'Close'],0)
+       if (quant > 0)
+         orders <- rbind(orders,
+               data.frame(order=c(-1,1,1),order.type=c(1,2,3),
+                          val = c(quant,
+                                  market[d,'Close']*(1-exp.prof),
+                                  market[d,'Close']*(1+max.loss)
+                                 ),
+                          action = c('open','close','close'),
+                          posID = c(NA,NA,NA)
+                         )
+                        )
+     }
+
+     # Now lets check if we need to close positions
+     # because their holding time is over
+     if (nOs)
+       for(i in 1:nOs) {
+         if (d - opened.pos[i,'Odate'] >= hold.time)
+           orders <- rbind(orders,
+                 data.frame(order=-opened.pos[i,'pos.type'],
+                            order.type=1,
+                            val = NA,
+                            action = 'close',
+                            posID = rownames(opened.pos)[i]
+                           )
+                          )
+       }
+
+     orders
+   }
```

函数 policy.1() 实现了 3.5.1 节中所描述的第一条交易策略。该函数有四个参数用来调整这个策略。这四个参数分别是：参数 bet 指明我们每次开新仓位时所投资的金额占当前资金的百分比；参数 exp.prof 表明我们所期望的当前仓位的收益率，当我们执行限价指令的时候会用到该参数；参数 max.loss 表明平仓前我们所能承受的最大损失，该参数用于止损指令；参数 hold.time 表明为实现指定收益率，我们愿意等待的天数。如果等待了 hold.time 天，仍然没有得到想要的收益率，该仓位将被平仓。

注意，每当我们建立一个新的仓位时，我们给模拟器发送三条指令：一个当前市场价开立新的仓位指令，一个限价指令来指明我们的目标收益率和一个止损指令限制我们的损失。

同样，以下函数实现我们的第二个交易策略：

```
> policy.2 <- function(signals,market,opened.pos,money,
+                      bet=0.2,exp.prof=0.025, max.loss= 0.05
+                     )
+   {
+     d <- NROW(market) # this is the ID of today
+     orders <- NULL
+     nOs <- NROW(opened.pos)
+     # nothing to do!
+     if (!nOs && signals[d] == 'h') return(orders)
+
+     # First lets check if we can open new positions
+     # i) long positions
+     if (signals[d] == 'b') {
+       quant <- round(bet*money/market[d,'Close'],0)
+       if (quant > 0)
+         orders <- rbind(orders,
+               data.frame(order=c(1,-1,-1),order.type=c(1,2,3),
+                          val = c(quant,
+                                  market[d,'Close']*(1+exp.prof),
+                                  market[d,'Close']*(1-max.loss)
+                                 ),
+                          action = c('open','close','close'),
+                          posID = c(NA,NA,NA)
+                         )
+                        )
+
+     # ii) short positions
+     } else if (signals[d] == 's') {
+       # this is the money already committed to buy stocks
+       # because of currently opened short positions
+       need2buy <- sum(opened.pos[opened.pos[,'pos.type']==-1,
+                                  "N.stocks"])*market[d,'Close']
+       quant <- round(bet*(money-need2buy)/market[d,'Close'],0)
+       if (quant > 0)
+         orders <- rbind(orders,
+               data.frame(order=c(-1,1,1),order.type=c(1,2,3),
+                          val = c(quant,
+                                  market[d,'Close']*(1-exp.prof),
+                                  market[d,'Close']*(1+max.loss)
+                                 ),
+                          action = c('open','close','close'),
+                          posID = c(NA,NA,NA)
+                         )
+                        )
+     }
+
+     orders
+   }
```

这个函数与之前的第一个交易策略函数非常相似。它们的主要差异在于：该交易策略允许在同一时间开立多个仓位，也没有限制平仓的时间。

定义了交易策略函数之后，我们就准备好了来尝试交易模拟器。为了演示方便，我们将选择数据集中的一个较小数据子集，建立支持向量机模型，然后用该模型来获取之后一个时间段的预测值。在某个交易策略下，用预测值来调用交易模拟器，得到利用支持向量机的交易信号所获得的交易结果。代码如下：

134 ~ 136

```
> # Train and test periods
> start <- 1
> len.tr <- 1000
> len.ts <- 500
> tr <- start:(start+len.tr-1)
> ts <- (start+len.tr):(start+len.tr+len.ts-1)
> # getting the quotes for the testing period
> data(GSPC)
> date <- rownames(Tdata.train[start+len.tr,])
> market <- GSPC[paste(date,'/',sep='')][1:len.ts]
> # learning the model and obtaining its signal predictions
> library(e1071)
> s <- svm(Tform,Tdata.train[tr,],cost=10,gamma=0.01)
> p <- predict(s,Tdata.train[ts,])
> sig <- trading.signals(p,0.1,-0.1)
> # now using the simulated trader
> t1 <- trading.simulator(market,sig,
+              'policy.1',list(exp.prof=0.05,bet=0.2,hold.time=30))
```

请注意，要运行以上代码，你必须提前创建用于建模的数据对象，具体方法请参见3.3.3节。

在调用交易模拟器时，我们采用第一种交易策略，提供了一些不同的值给它的某些参数。我们使用默认的交易成本（5个货币单位）、默认的初始资本（100万货币单位）。指令的结果是一个 tradeRecord 类对象。我们可以检查该返回对象的内容，如下所示：

```
> t1

Object of class tradeRecord with slots:

        trading: <xts object with a numeric  500 x 5  matrix>
        positions: <numeric  16 x 7  matrix>
        init.cap :  1e+06
        trans.cost :  5
        policy.func :  policy.1
        policy.pars : <list with  3  elements>

> summary(t1)

== Summary of a Trading Simulation with  500  days ==

Trading policy function :  policy.1
Policy function parameters:
        exp.prof  =  0.05
        bet  =  0.2
        hold.time  =  30

Transaction costs :  5
Initial Equity     :  1e+06
Final Equity       :  997211.9   Return :  -0.28 %
Number of trading positions:  16

Use function "tradingEvaluation()" for further stats on this simulation.
```

函数 tradingEvaluation( ) 用于获得在模拟交易期间的一系列表示交易效果的经济指标：

```
> tradingEvaluation(t1)

   NTrades        NProf     PercProf          PL          Ret    RetOverBH
     16.00         8.00        50.00    -2788.09        -0.28        -7.13
     MaxDD  SharpeRatio      AvgProf     AvgLoss        AvgPL      MaxProf
  59693.15         0.00         4.97       -4.91         0.03         5.26
   MaxLoss
     -5.00
```

也可以使用函数 plot( )，绘制一个交易效果的概览图：

```
> plot(t1, market, theme = "white", name = "SP500")
```

该命令的结果如图 3-6 所示。

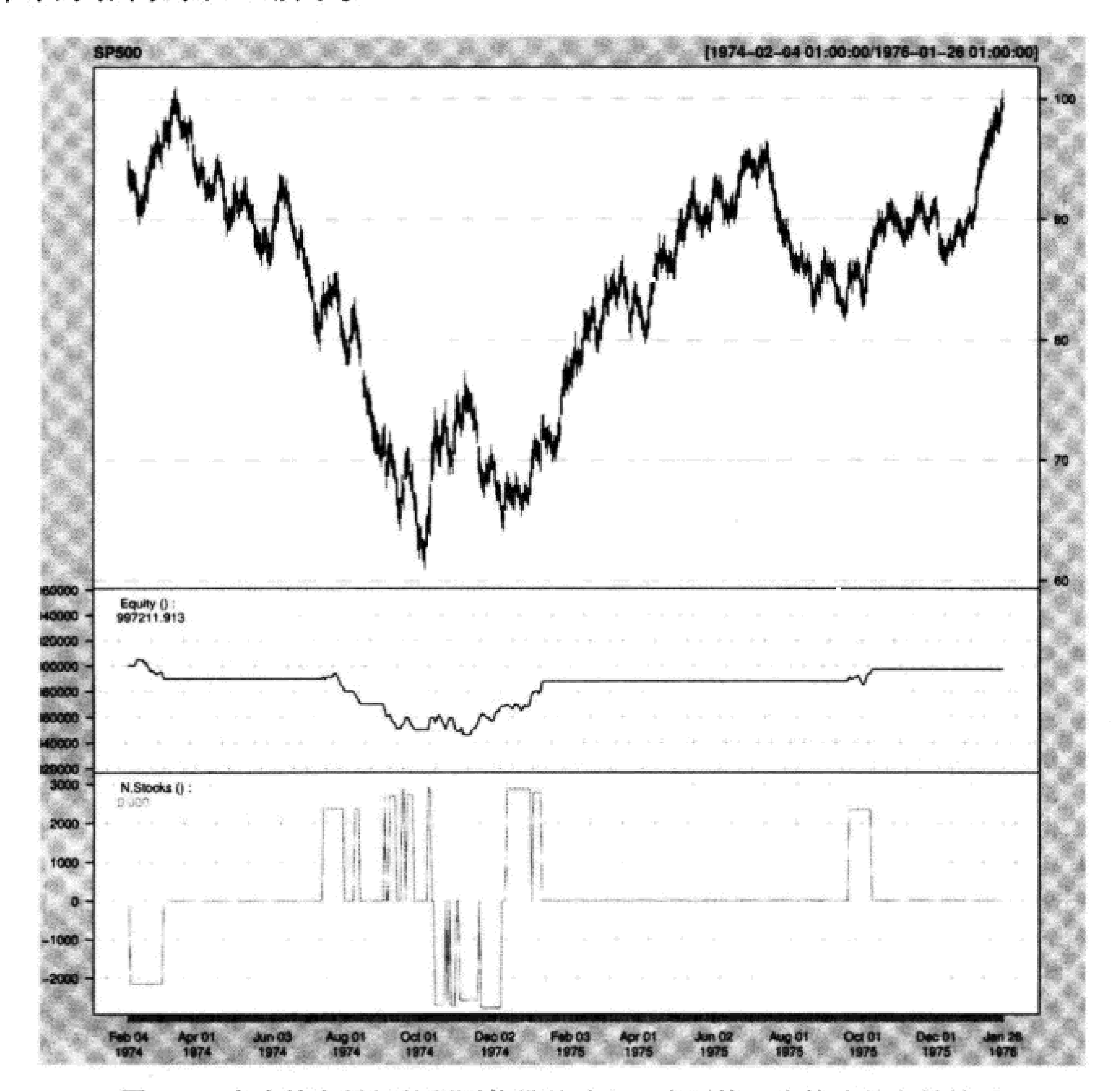


图 3-6　在支持向量机的预测信号基础上，应用第一个策略的交易结果

图 3-6 显示这个交易的效果不好，收益率为负值。如果采用第二个交易策略，交易结果会不同吗？看看下面的代码：

```
> t2 <- trading.simulator(market, sig, "policy.2", list(exp.prof = 0.05,
+      bet = 0.3))
> summary(t2)

== Summary of a Trading Simulation with  500  days ==

Trading policy function :  policy.2
Policy function parameters:
```

```
        exp.prof  =  0.05
        bet  =  0.3

Transaction costs :  5
Initial Equity     :  1e+06
Final Equity       :  961552.5   Return :  -3.84 %
Number of trading positions:  29

Use function "tradingEvaluation()" for further stats on this simulation.

> tradingEvaluation(t2)
  NTrades       NProf    PercProf          PL         Ret   RetOverBH
    29.00       14.00       48.28   -38447.49       -3.84      -10.69
    MaxDD SharpeRatio    AvgProf     AvgLoss       AvgPL     MaxProf
156535.05       -0.02        4.99       -4.84       -0.10        5.26
  MaxLoss
    -5.00
```

使用相同的交易信号，但应用不同的交易策略，收益率从 -0.27% 下降到了 -2.86%。让我们用一个不同的训练和测试时间段再做一次上述的实验：

```
> start <- 2000
> len.tr <- 1000
> len.ts <- 500
> tr <- start:(start + len.tr - 1)
> ts <- (start + len.tr):(start + len.tr + len.ts - 1)
> s <- svm(Tform, Tdata.train[tr, ], cost = 10, gamma = 0.01)
> p <- predict(s, Tdata.train[ts, ])
> sig <- trading.signals(p, 0.1, -0.1)
> t2 <- trading.simulator(market, sig, "policy.2", list(exp.prof = 0.05,
+     bet = 0.3))
> summary(t2)

== Summary of a Trading Simulation with  500  days ==

Trading policy function :  policy.2
Policy function parameters:
        exp.prof  =  0.05
        bet  =  0.3

Transaction costs :  5
Initial Equity     :  1e+06
Final Equity       :  107376.3   Return :  -89.26 %
Number of trading positions:  229

Use function "tradingEvaluation()" for further stats on this simulation.

> tradingEvaluation(t2)

  NTrades       NProf    PercProf          PL         Ret   RetOverBH
   229.00       67.00       29.26  -892623.73      -89.26      -96.11
    MaxDD SharpeRatio    AvgProf     AvgLoss       AvgPL     MaxProf
959624.80       -0.08        5.26       -4.50       -1.65        5.26
  MaxLoss
    -5.90
```

137~140 这次模拟交易应用了相同的建模技术和相同的交易策略，获得了相当糟糕的结果。这里得到的主要经验是：需要可靠的统计估计。不要被少数几次的重复实验结果所愚弄，即使测试期区间为两年也是不够的。我们需要不同条件下更多的重复次数，以确保所得到的结果在统计上是

可靠的。对于时间序列模型尤其如此，这类模型的不同时期可能有不同的模式（例如，不同的时期有不同的波动率或不同的趋势）。这是3.6节将要讨论的问题。

3.6 模型评价和选择

本节将学习如何获取模型交易效果评价指标的可靠估计，这些指标的估计值可以使我们合理地对多个不同的交易系统进行比较和选择。

3.6.1 蒙特卡罗估计

时间序列问题，例如我们正在处理的问题，给获取可靠的模型评价指标的估计值带来新的挑战。这是由于所有的时间序列数据观察值都附有一个时间标签，这个时间标签给出了数据的一个内在顺序。这个顺序需要特别注意，以防止出现不可靠估计值的风险。在第2章，我们用交叉验证的方法来获取评价指标的可靠估计值。这种方法包括了一个随机重新抽样步骤来改变原始观测值的顺序。这意味着交叉验证方法不适用于时间序列问题。采用这种方法意味着用于模型测试的观测值可能比用于建立模型的训练集数据还要时间久远。在现实中这是不可行的，因此通过这个过程获取的估计值是不可靠的并且可能过于乐观。因为给定未来的观测值会更容易地预测过去的观测值，反之则不然。

运用时间序列数据的估计过程中应当确保用于模型测试的数据比建立模型的数据新。这意味着不能对观测值采用随机重新抽样的方法或者其他可能改变时间序列数据的时间标签的过程。然而，任何合理的估计过程应当包括一些随机选择过程，以确保所获取估计值的统计可靠性。这包括了在不同条件下多次重复估计过程，最好包含随机选择过程。给定一个时间序列数据，其时间区间从时间 $t \sim t+N$，我们如何实现带有随机选择的重复估计过程？首先，我们需选择用于模型估计的训练集和测试集数据。这就要决定用于模型估计过程的训练集和测试集数据的大小。这两个数据集的大小之和应小于 N，这样才能确保我们可以从已知数据集中随机选择用于重复实验的不同数据集。然而，如果我们选择太小的训练集，它就可能严重影响模型的性能。同样，太小的测试集可能会导致模型不稳定。特别是如果我们怀疑时间序列中存在模式变化，我们希望在这种模式变化情况下测试模型，测试集太小将导致问题。 141

我们的数据集包括大约30年的每日报价。这里对所有模型都设置测试集为5年的日报价数据，训练集为10年的日报价数据。这种设置确保了训练集和测试集充分大。而且，由于我们有30年的日报价数据，所以这种样本量的选择就为测试过程重复不同的设置留下了空间。

在实验方法上，我们选择蒙特卡罗实验来获取模型评估指标的可靠估计。蒙特卡罗方法依靠随机取样来获取估计结果。我们将用这个取样过程在30年日报价数据中选择一个由 R 个数据点构成的集合。对于集合中的每一个随机选取的时间点 r，我们用这个时间点之前10年的日报价数据来获取模型并用这个点之后5年的数据测试这些模型。在进行 R 次迭代之后，我们将得到每个性能评估指标的 R 个估计值。每一个指标估计值都是通过随机选取的15年数据窗口得到的，前10年用做训练模型，后5年用做测试模型。这种设置确保了时间序列数据的时间排序。重复这个过程 R 次，将确保有充分变化的训练和测试条件，这就增加了估计值的可靠性。而且，如果我们在评估不同模型时用同样一组随机选取的 R 个数据点，那么就能够进行配对比较，从而得到不同模型的平均性能之差的统计置信度水平。图3-7总结了上面描述的蒙特卡罗实验方法，注意对每一个随机点 r，必须确保它之前有10年数据，它之后有5年的数据，这就使某些数据点被排除在随机选择的 R 个数据点之外。

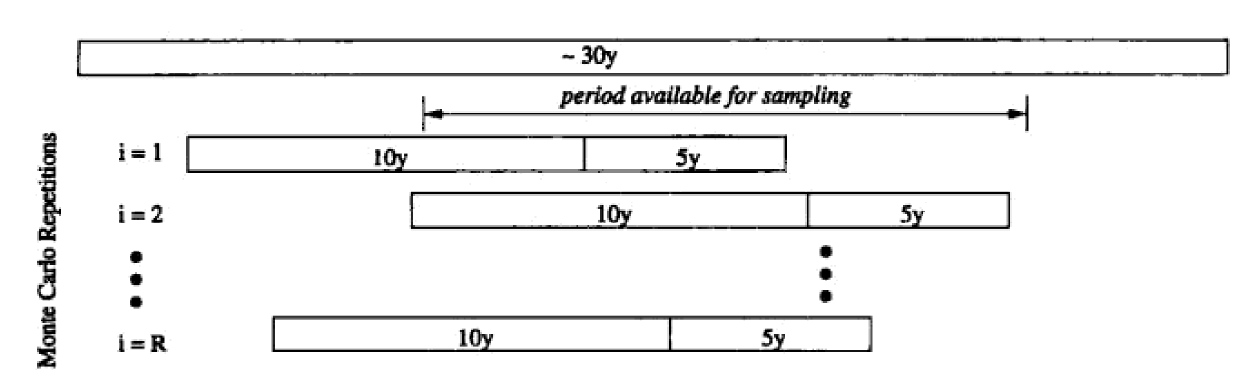


图 3-7 蒙特卡罗实验过程

第 2 章中用于进行 k 折交叉验证实验的函数 experimentalComparison( )，也可以用于蒙特卡罗实验。在 3.6.2 节，我们将使用这个函数获取多个交易系统性能评价指标的可靠估计。

142

3.6.2 实验比较

本节描述一组蒙特卡罗实验，它们用于获取在 3.3.4 节和 3.5.2 节提到的模型性能评价指标的可靠估计。用于这些实验的数据是在 3.3.3 节结尾部分生成的数据集。

在这些实验中所考虑每一个模型都会使用三个不同的模型更新设置。这些更新方式已经在 3.4.1 节描述过，它们是应用于所有 5 年测试集的单一模型、滑动窗口或者增长窗口。本书有两个函数可以使用具有任何窗口模式的模型。函数 slidingWindow( ) 和函数 growingWindow( ) 都有 5 个主要参数：第一个参数是 learner 类对象，在第 2 章我们应用过该类对象，它用来保存模型的所有细节（函数名和参数值）；第二个参数是描述预测任务的公式；第三个参数和第四个参数分别设置训练集和测试集数据；第五个参数是窗口模式所应用的重训练步骤，在这个参数指定测试个案的数量之后，将对刚刚获得的模型所应用的训练集数据进行滑动或者增长，然后重新训练模型。两个函数都使用相应的窗口模式返回测试集的模型预测值。

下面的代码创建了一组函数，它们用于执行比较不同交易系统的整个"训练 + 测试 + 评估"过程周期。按照图 3-7 所示的蒙特卡罗实验模式，蒙特卡罗过程将在不同的"训练 + 测试"时期中调用这些函数。代码如下：

```
> MC.svmR <- function(form, train, test, b.t = 0.1, s.t = -0.1,
+     ...) {
+     require(e1071)
+     t <- svm(form, train, ...)
+     p <- predict(t, test)
+     trading.signals(p, b.t, s.t)
+ }
> MC.svmC <- function(form, train, test, b.t = 0.1, s.t = -0.1,
+     ...) {
+     require(e1071)
+     tgtName <- all.vars(form)[1]
+     train[, tgtName] <- trading.signals(train[, tgtName],
+         b.t, s.t)
+     t <- svm(form, train, ...)
+     p <- predict(t, test)
+     factor(p, levels = c("s", "h", "b"))
+ }
> MC.nnetR <- function(form, train, test, b.t = 0.1, s.t = -0.1,
+     ...) {
+     require(nnet)
+     t <- nnet(form, train, ...)
+     p <- predict(t, test)
```

```
+     trading.signals(p, b.t, s.t)
+ }
> MC.nnetC <- function(form, train, test, b.t = 0.1, s.t = -0.1,
+     ...) {
+     require(nnet)
+     tgtName <- all.vars(form)[1]
+     train[, tgtName] <- trading.signals(train[, tgtName],
+         b.t, s.t)
+     t <- nnet(form, train, ...)
+     p <- predict(t, test, type = "class")
+     factor(p, levels = c("s", "h", "b"))
+ }
> MC.earth <- function(form, train, test, b.t = 0.1, s.t = -0.1,
+     ...) {
+     require(earth)
+     t <- earth(form, train, ...)
+     p <- predict(t, test)
+     trading.signals(p, b.t, s.t)
+ }
> single <- function(form, train, test, learner, policy.func,
+     ...) {
+     p <- do.call(paste("MC", learner, sep = "."), list(form,
+         train, test, ...))
+     eval.stats(form, train, test, p, policy.func = policy.func)
+ }
> slide <- function(form, train, test, learner, relearn.step,
+     policy.func, ...) {
+     real.learner <- learner(paste("MC", learner, sep = "."),
+         pars = list(...))
+     p <- slidingWindowTest(real.learner, form, train, test,
+         relearn.step)
+     p <- factor(p, levels = 1:3, labels = c("s", "h", "b"))
+     eval.stats(form, train, test, p, policy.func = policy.func)
+ }
> grow <- function(form, train, test, learner, relearn.step,
+     policy.func, ...) {
+     real.learner <- learner(paste("MC", learner, sep = "."),
+         pars = list(...))
+     p <- growingWindowTest(real.learner, form, train, test,
+         relearn.step)
+     p <- factor(p, levels = 1:3, labels = c("s", "h", "b"))
+     eval.stats(form, train, test, p, policy.func = policy.func)
+ }
```

函数 MC. x( ) 应用所给的公式和训练集来获得不同的模型，并且用已给的测试集来测试这些模型，然后返回模型的预测值。如果可能，我们会有两个版本的模型：一个版本是回归任务（返回的模型名以字母“R”结尾），另一个版本是分类任务（返回的模型名以“C”结尾）。注意，这两个模型得到的最后预测信号的预处理和后处理步骤是不同的。这些函数被函数 single( )、函数 slide( ) 和函数 grow( ) 调用，这三个函数通过使用参数 learner 所指定的模型和相应的模型更新机制来获得测试集的预测值。在获取预测值之后，这些函数会调用下面的函数 eval. stats( ) 来得到想要估计的模型评价指标统计量。函数 eval. stats( ) 的实现如下：

143 ~ 144

```
> eval.stats <- function(form,train,test,preds,b.t=0.1,s.t=-0.1,...) {
+   # Signals evaluation
+   tgtName <- all.vars(form)[1]
```

```
+   test[,tgtName] <- trading.signals(test[,tgtName],b.t,s.t)
+   st <- sigs.PR(preds,test[,tgtName])
+   dim(st) <- NULL
+   names(st) <- paste(rep(c('prec','rec'),each=3),
+                      c('s','b','sb'),sep='.')
+
+   # Trading evaluation
+   date <- rownames(test)[1]
+   market <- GSPC[paste(date,"/",sep='')][1:length(preds),]
+   trade.res <- trading.simulator(market,preds,...)
+
+   c(st,tradingEvaluation(trade.res))
+ }
```

函数 eval. stats( ) 用其他两个函数来收集信号的决策精确度、回溯精确度，以及其他几个经济评价指标。函数 sigs. PR( ) 接收的参数是预测信号和真实信号，它分别计算卖出、买入和“卖出 + 买入”信号的决策精确度和回溯精确度。另一个函数是 tradingEvaluation( )，它用来获得给定交易记录的经济评价指标，而这个交易记录是用函数 trading. simulator( ) 获取的，该函数可用来按照模型信号在市场上进行模拟交易。以上所有这些函数已在 3. 5. 3 节中充分介绍过。

在蒙特卡罗程序中调用适当参数设置的函数 single( )、函数 slide( ) 和函数 grow( )，可以得到我们需要比较的模型。以下介绍如何建立一个循环程序，在循环中运行一系列的交易系统，并调用这些函数来获取这些交易系统的性能估计。每一个交易系统由一些具有特定参数的学习模型和交易策略构成。交易策略将指示交易中如何应用模型的预测信号。下面我们考虑三种交易策略，它们是从 3. 5. 3 节描述的策略（policy. 1( ) 和 policy. 2( )）衍生而来，以下函数实现这三个衍生策略：

```
> pol1 <- function(signals,market,op,money)
+   policy.1(signals,market,op,money,
+            bet=0.2,exp.prof=0.025,max.loss=0.05,hold.time=10)
> pol2 <- function(signals,market,op,money)
+   policy.1(signals,market,op,money,
+            bet=0.2,exp.prof=0.05,max.loss=0.05,hold.time=20)
> pol3 <- function(signals,market,op,money)
+   policy.2(signals,market,op,money,
+            bet=0.5,exp.prof=0.05,max.loss=0.05)
```

下列代码运行蒙特卡罗实验。这里建议你在运行下列代码前要慎重考虑。即使在速度相当快的计算机上，也需要几天才能运行完下列程序。在本书的网站上，我们提供了运行这个实验所得到的结果。所以你不用运行该实验也可以重复 3. 6. 3 节的结果分析。

```
> # The list of learners we will use
> TODO <- c('svmR','svmC','earth','nnetR','nnetC')
> # The datasets used in the comparison
> DSs <- list(dataset(Tform,Tdata.train,'SP500'))
> # Monte Carlo (MC) settings used
> MCsetts <- mcSettings(20,     # 20 repetitions of the MC exps
+                       2540,   # ~ 10 years for training
+                       1270,   # ~ 5 years for testing
+                       1234)   # random number generator seed
> # Variants to try for all learners
> VARS <- list()
> VARS$svmR    <- list(cost=c(10,150),gamma=c(0.01,0.001),
+                      policy.func=c('pol1','pol2','pol3'))
> VARS$svmC    <- list(cost=c(10,150),gamma=c(0.01,0.001),
```

```
+                       policy.func=c('pol1','pol2','pol3'))
> VARS$earth <- list(nk=c(10,17),degree=c(1,2),thresh=c(0.01,0.001),
+                      policy.func=c('pol1','pol2','pol3'))
> VARS$nnetR  <- list(linout=T,maxit=750,size=c(5,10),
+                       decay=c(0.001,0.01),
+                       policy.func=c('pol1','pol2','pol3'))
> VARS$nnetC  <- list(maxit=750,size=c(5,10),decay=c(0.001,0.01),
+                       policy.func=c('pol1','pol2','pol3'))
> # main loop
> for(td in TODO) {
+   assign(td,
+          experimentalComparison(
+            DSs,
+            c(
+              do.call('variants',
+                      c(list('single',learner=td),VARS[[td]],
+                        varsRootName=paste('single',td,sep='.'))),
+              do.call('variants',
+                      c(list('slide',learner=td,
+                             relearn.step=c(60,120)),
+                        VARS[[td]],
+                        varsRootName=paste('slide',td,sep='.'))),
+              do.call('variants',
+                      c(list('grow',learner=td,
+                             relearn.step=c(60,120)),
+                        VARS[[td]],
+                        varsRootName=paste('single',td,sep='.')))
+              ),
+            MCsetts)
+          )
+
+   # save the results
+   save(list=td,file=paste(td,'Rdata',sep='.'))
+ }
```

MCsetts 对象控制实验的总体参数，它给出重复的次数（20）、训练集的大小（2540 天 ~ 10 年）、测试集的大小（1270 天 ~ 5 年）、所用的随机数生成器的种子。

列表 VARS 给出了我们将要实验的每个模型的所有参数变化，列表中给出参数的所有可能的组合就是所有可能的参数变化。在 3 个不同的模型更新模式（单窗口、滑动窗口和增广窗口）中运行每一个参数变化。此外，对于后面两个模型更新模式，我们会尝试两个重新训练步骤：60 天和 120 天。

对于 svm 模型，我们实验了 4 个训练参数变化和 3 个不同的交易策略，就有 12 个模型变体；对于 earth 模型，实验了 24 个模型变体；对于 nnet 模型，也实验了 12 个模型变体。每一个模型变体将在单个模式及 4 个窗口模式（两个策略和两个不同的重复训练步骤）下运行。显然，这将导致需要进行大量的实验。即，一共有 60（ = 12 + 24 + 24）个 svm 模型变体、120（ = 24 + 48 + 48）个 earth 模型变体和 60 个 nnet 模型变体。它们中的每一个都将对 10 年的训练集和 5 年的测试集重复执行 20 次。这就是为什么我们说运行这个实验需要花很长时间。然而，在描述这个问题的时候，我们提到这里的实验仅仅是所有解决方法中的一个小例子。其中有太多的“小”决定，将导致我们有其他不同的实验方式（如买入和卖出的界限值以及其他学习系统等）。这意味着在本案例的应用领域，任何正式的尝试都需要大量的计算资源来进行合适的模型选择，显然这在本书的讨论范畴之外。这里的目的是给读者提供适合的方法指导而不是对这里的特定数据找出最好的交易系统。

145 ~ 147

3.6.3 结果分析

3.6.2 节提供的代码生成了 5 个数据文件，这些文件中的对象含有实验的 5 个训练系统所有模型变体的实验结果。这些数据文件被命名为 “svmR. Rdata”、“svmC. Rdata”、“earth. Rdata”、“nnetR. Rdata” 和 “nnetC. Rdata”。每一个文件都包含与文件相同名字（除文件扩展名外）的对象。这些对象是 compExp 类对象，本书添加包中含有探索这些对象所存储结果的多种方法。

也许你没有亲自运行这个实验，那么你可在本书网站上找到这 5 个数据文件，将它们下载到计算机中，并运行以下代码将这些对象载入 R 软件。

```
> load("svmR.Rdata")
> load("svmC.Rdata")
> load("earth.Rdata")
> load("nnetR.Rdata")
> load("nnetC.Rdata")
```

对于每一个交易系统变体，我们测量了多个性能指标。有些性能指标用来衡量信号预测的正确性，另一些指标用来衡量应用这些信号进行交易时的经济效果。根据对实验中得到这些性能指标的综合考虑，决定哪一个模型是最好的。最终选定的模型可能取决于我们最注重的那个性能指标。

尽管性能评价的指标有多种，但是其中的某些指标更具有相关性。在我们的案例中，在衡量预测信号正确性的评价指标中，预测精确度指标比回溯精确度指标更重要。实际上，预测精确度和预测信号相关，而预测信号将决定是否开立新的仓位等交易行为。低预测精确度是因为预测信号错误，它意味着在错误的时间开立仓位。这显然会导致极大的损失。而回溯精确度则没有这种潜在的损失，它衡量模型捕获交易机会的能力。如果回溯精确度取值较低，就意味着机会的错失，但并不意味着高损失。因此，我们对模型的 “prec. sb” 统计量特别感兴趣，它衡量买入信号和卖出信号的预测精确度。

对交易效果衡量而言，交易系统的回报很重要（实验中的统计量 “Ret”），买入并持有策略的回报（实验中的统计量 “RetOverBH”）也很重要。盈利交易所占百分比也同样重要，显然它应该在 50%（统计量 “PercProf”）以上。在衡量风险分析方面，这里考虑夏普比率值（“Sharp”）和最大回撤值（“MaxDD”）。

函数 summary( ) 可以用于已经载入的 compExp 对象。然而，考虑到很大数量的模型变体和性能指标统计量，输出结果的内容是相当多的。

148 另一种方式是使用本书添加包提供的函数 rankSystems( )。使用这个函数，可以获得感兴趣的评价指标的最优取值，它指出最好的模型以及相应的评价指标值。代码如下：

```
> tgtStats <- c('prec.sb','Ret','PercProf',
+               'MaxDD','SharpeRatio')
> allSysRes <- join(subset(svmR,stats=tgtStats),
+                   subset(svmC,stats=tgtStats),
+                   subset(nnetR,stats=tgtStats),
+                   subset(nnetC,stats=tgtStats),
+                   subset(earth,stats=tgtStats),
+                   by = 'variants')
> rankSystems(allSysRes,5,maxs=c(T,T,T,F,T))

$SP500
$SP500$prec.sb
          system score
```

```
1  slide.svmC.v5      1
2  slide.svmC.v6      1
3 slide.svmC.v13      1
4 slide.svmC.v14      1
5 slide.svmC.v21      1

$SP500$Ret
             system   score
1 single.nnetR.v12 97.4240
2  single.svmR.v11  3.4960
3  slide.nnetR.v15  2.6230
4  single.svmC.v12  0.7875
5   single.svmR.v8  0.6115

$SP500$PercProf
         system   score
1 grow.nnetR.v5 60.4160
2 grow.nnetR.v6 60.3640
3 slide.svmR.v3 60.3615
4  grow.svmR.v3 59.8710
5 grow.nnetC.v1 59.8615

$SP500$MaxDD
           system    score
1  slide.svmC.v5 197.3945
2  slide.svmC.v6 197.3945
3   grow.svmC.v5 197.3945
4   grow.svmC.v6 197.3945
5 slide.svmC.v13 399.2800

$SP500$SharpeRatio
           system score
1  slide.svmC.v5  0.02
2  slide.svmC.v6  0.02
3 slide.svmC.v13  0.02
4 slide.svmC.v14  0.02
5 slide.svmC.v21  0.02
```

函数 subset() 可以应用于 compExps 对象，选择存储在这些对象中的部分信息。这时，我们只选择估计出的性能指标的子集。然后，我们使用函数 join() 将所有的模型变体合在一起，存储在一个 compExps 对象中。这个函数可以把具有不同维数的 compExps 对象结合在一起。本案例中模型变体的其他实验条件是一样的，因此这样结合是有意义的。最后，我们运用函数 rankSystems() 从我们所有的交易系统的性能指标中选择出分值最高的 5 个。最好的性能指标值随指标不同而变化。有时我们需要最高的指标值，有时却需要最低的指标值。这可以用函数 rankSystems() 中的参数 maxs 来设定，它可以指定哪个指标需要最大化的取值。

我们观察这 5 个最优性能指标，发现它们或者采用 svm 算法，或者采用 nnet 算法。另一个显而易见的模式是，几乎所有的这些模型变体都运用了某个窗口机制。这提供了窗口机制优于单一模型的某种证据，也可以认为它确认了这些数据有模式变化。我们也可以观察到多个显著的（或可疑的）分数，即买入或者卖出信号的预测精确度。获得 100% 的预测精确度是很奇怪的。仔细检查这些交易系统的结果，它揭示得到这么高的分数，是由于在 5 年测试期中有极少的信号。

```
> summary(subset(svmC,
+                stats=c('Ret','RetOverBH','PercProf','NTrades'),
+                vars=c('slide.svmC.v5','slide.svmC.v6')))

== Summary of a  Monte Carlo  Experiment ==

 20  repetitions Monte Carlo Simulation using:
        seed =  1234
        train size =  2540  cases
        test size =  1270  cases

* Datasets ::  SP500
* Learners  ::  slide.svmC.v5, slide.svmC.v6

* Summary of Experiment Results:

-> Datataset:  SP500

        *Learner: slide.svmC.v5
              Ret  RetOverBH  PercProf   NTrades
avg     0.0250000  -77.10350   5.00000 0.0500000
std     0.1118034   33.12111  22.36068 0.2236068
min     0.0000000 -128.01000   0.00000 0.0000000
max     0.5000000  -33.77000 100.00000 1.0000000
invalid 0.0000000    0.00000   0.00000 0.0000000

        *Learner: slide.svmC.v6
              Ret  RetOverBH  PercProf   NTrades
avg     0.0250000  -77.10350   5.00000 0.0500000
std     0.1118034   33.12111  22.36068 0.2236068
min     0.0000000 -128.01000   0.00000 0.0000000
max     0.5000000  -33.77000 100.00000 1.0000000
invalid 0.0000000    0.00000   0.00000 0.0000000
```

事实上，这些方法在整个测试期内至多进行了一次交易，其平均收益为 0.25%，这比简单的买入并持有策略的收益低 -77.1%，这些显然是无用的模型。

对整体排名的最后一个评论是，就最大回撤而言，不能认为模型太坏，然而夏普比率值是绝对令人失望的。

为了得出对所有这些模型变体的结论，需要对这些模型性能统计指标增加一些约束条件。我们假设以下最小值：1）一个合理的平均交易数量，例如需超过 20；2）平均回报至少大于 0.5%（考虑到这些交易系统平均回报的取值较低，因此设为该值）；3）盈利交易的百分比大于 40%。现在检查一下是否有交易系统满足这些约束条件。

```
> fullResults <- join(svmR, svmC, earth, nnetC, nnetR, by = "variants")
> nt <- statScores(fullResults, "NTrades")[[1]]
> rt <- statScores(fullResults, "Ret")[[1]]
> pp <- statScores(fullResults, "PercProf")[[1]]
> s1 <- names(nt)[which(nt > 20)]
> s2 <- names(rt)[which(rt > 0.5)]
> s3 <- names(pp)[which(pp > 40)]
> namesBest <- intersect(intersect(s1, s2), s3)

> summary(subset(fullResults,
               stats=tgtStats,
               vars=namesBest))
```

```
== Summary of a  Monte Carlo  Experiment ==

 20  repetitions Monte Carlo Simulation using:
         seed =  1234
         train size =  2540  cases
         test size =  1270  cases
* Datasets ::  SP500
* Learners  ::  single.nnetR.v12, slide.nnetR.v15, grow.nnetR.v12

* Summary of Experiment Results:


-> Dataset:  SP500
        *Learner: single.nnetR.v12
           prec.sb       Ret  PercProf      MaxDD SharpeRatio
avg     0.12893147   97.4240 45.88600  1595761.4 -0.01300000
std     0.06766129  650.8639 14.04880  2205913.7  0.03798892
min     0.02580645 -160.4200 21.50000   257067.4 -0.08000000
max     0.28695652 2849.8500 73.08000 10142084.7  0.04000000
invalid 0.00000000    0.0000  0.00000        0.0  0.00000000

        *Learner: slide.nnetR.v15
           prec.sb      Ret  PercProf    MaxDD SharpeRatio
avg     0.14028491  2.62300 54.330500 46786.28  0.01500000
std     0.05111339  4.93178  8.339434 23526.07  0.03052178
min     0.03030303 -7.03000 38.890000 18453.94 -0.04000000
max     0.22047244  9.85000 68.970000 99458.44  0.05000000
invalid 0.00000000  0.00000  0.000000     0.00  0.00000000

        *Learner: grow.nnetR.v12
           prec.sb        Ret  PercProf     MaxDD SharpeRatio
avg     0.18774920   0.544500 52.66200  41998.26  0.00600000
std     0.07964205   4.334151 11.60824  28252.05  0.03408967
min     0.04411765 -10.760000 22.22000  18144.11 -0.09000000
max     0.33076923   5.330000 72.73000 121886.17  0.05000000
invalid 0.00000000   0.000000  0.00000      0.00  0.00000000
```

为了获取满足上述约束条件的交易系统变体的名字，我们使用了本书添加包中的 statScores( ) 函数。该函数的参数包括 compExp 对象以及性能指标的名字，在默认情况下，该函数提供所有交易系统在这些性能指标上的平均值。该函数的结果是一个列表，列表的元素个数和实验中数据集的个数相同（在我们的实验中是一个单一数据集）。用户可以在函数 statScores( ) 的第三个可选参数中指定一个函数来获取另一个数值汇总量，而不是平均值。使用这个函数获得的结果，将得到满足每一个约束条件的交易系统的名称。运用函数 intersect( ) （它是多个指标值集合的交集），就可以获得满足所有约束条件的交易系统的名称。

正如我们看到的，在我们比较的 240 个交易系统变体中只有 3 个交易系统满足这些最小约束。所有这 3 个系统都应用回归任务并且都是基于神经网络模型。这 3 个模型以不同的方式来应用训练集数据。方法“single. nnetR. v12”没有应用任何窗口机制，并且获得了极高的平均回报值 97.4%。然而，如果我们仔细观察这个交易系统的结果，就会发现在其中的一次迭代，该系统的回报很低，为 -160.40%。显然，该系统的结果是极其不稳定的，它的标准差很大，为 650.86%，这也验证了该系统结果的不稳定性。另外两个系统的得分相似。下面代码运用函数 compAnalysis( ) 来对结果进行显著性统计分析。

149 ~ 152

```
> compAnalysis(subset(fullResults,
+                     stats=tgtStats,
+                     vars=namesBest))


== Statistical Significance Analysis of Comparison Results ==

Baseline Learner::          single.nnetR.v12  (Learn.1)

** Evaluation Metric::          prec.sb

- Dataset: SP500
       Learn.1    Learn.2 sig.2    Learn.3 sig.3

AVG 0.12893147 0.14028491        0.18774920    +
STD 0.06766129 0.05111339        0.07964205

** Evaluation Metric::          Ret

- Dataset: SP500
     Learn.1 Learn.2 sig.2  Learn.3 sig.3
AVG  97.4240 2.62300     -  0.544500    -
STD 650.8639 4.93178        4.334151

** Evaluation Metric::          PercProf

- Dataset: SP500
     Learn.1   Learn.2 sig.2  Learn.3 sig.3
AVG 45.88600 54.360500     +  52.66200
STD 14.04880  8.339434        11.60824

** Evaluation Metric::          MaxDD

- Dataset: SP500
    Learn.1  Learn.2 sig.2  Learn.3 sig.3
AVG 1595761 46786.28    -- 41998.26    --
STD 2205914 23526.07       28252.05

** Evaluation Metric::          SharpeRatio

- Dataset: SP500
        Learn.1    Learn.2 sig.2    Learn.3 sig.3
AVG -0.01300000 0.01500000     +  0.00600000
STD  0.03798892 0.03052178        0.03408967

Legends:
Learners -> Learn.1 = single.nnetR.v12 ; Learn.2 = slide.nnetR.v15 ;
            Learn.3 = grow.nnetR.v12 ;
Signif. Codes -> 0 '++' or '--' 0.001 '+' or '-' 0.05 ' ' 1
```

注意以上代码会生成一些警告，那是由于某些系统不能获得某些性能指标的有效值所导致的（例如，没有买入或者卖出信号将导致无效的决策精确度得分）。

尽管结果存在变化性，但以上 Wilcoxon 检验告诉我们，交易系统“single. nnetR. v12”的平均回报以 95% 的置信度大于其他交易系统。然而，就其他性能指标而言，这个系统明显比其他系统差。

通过绘制 compExp 对象，可以较好地了解这些性能指标在 20 次重复实验中的分布情况。代码如下：

```
> plot(subset(fullResults,
+             stats=c('Ret','PercProf','MaxDD'),
+             vars=namesBest))
```

以上代码的结果如图 3-8 所示。

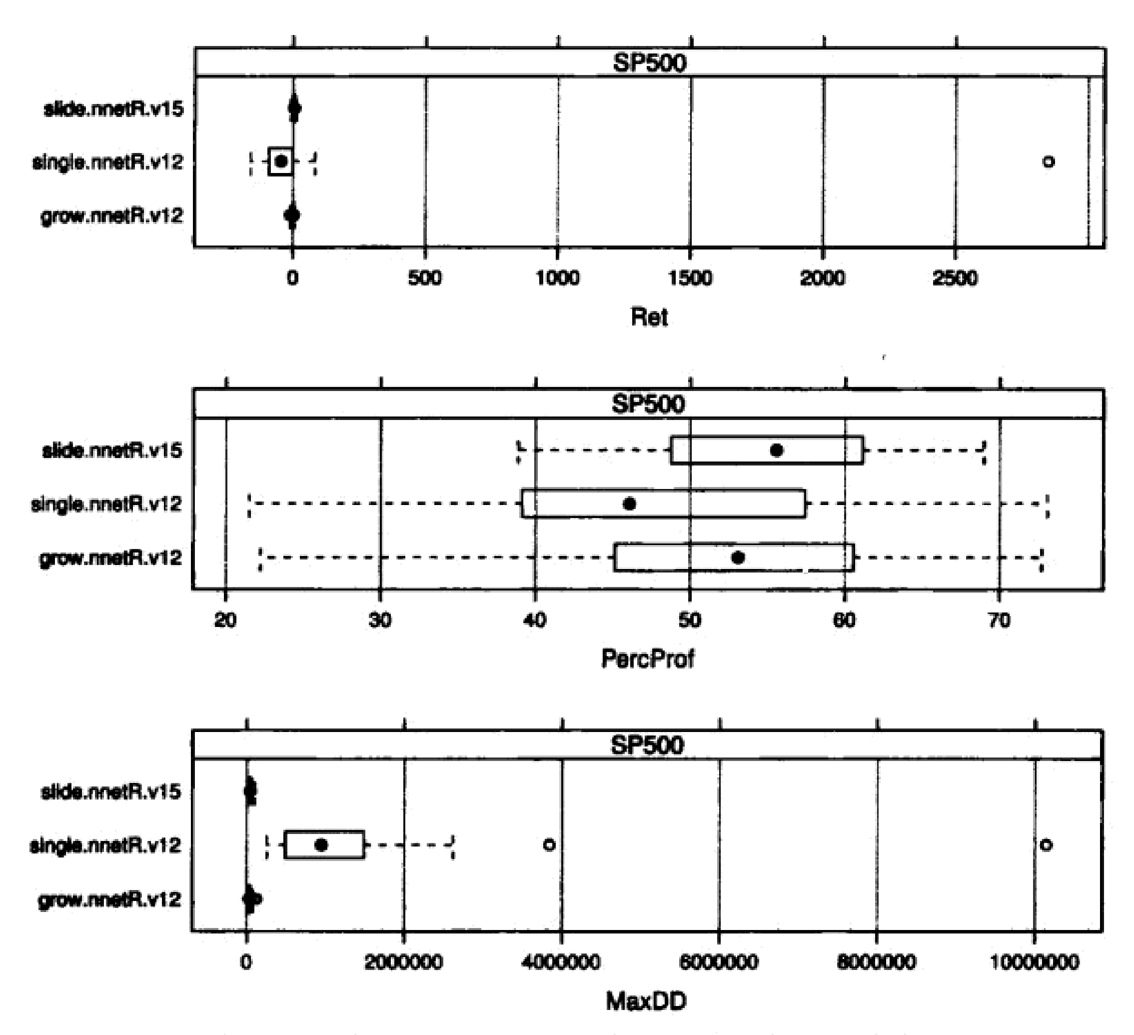


图 3-8　最好的 3 个交易系统 20 次实验的性能指标值

两个应用窗口机制的交易系统的性能指标值相似，这使得对它们的区分比较困难。相反，"single. nnetR. v12" 的结果却明显和上面两个系统不一样。我们可以发现，它之所以能有较高的平均回报，原因是其中的一个蒙特卡罗实验有明显异常的回报值（大约为 2800%）。这个系统的其他指标的估计值看起来要明显比另外两个系统差。为了满足我们的好奇，可以运用函数 getVariant() 检查这个特定交易系统的设置情况，代码如下：

```
> getVariant("single.nnetR.v12", nnetR)

Learner::  "single"

Parameter values
         learner  =  "nnetR"
         linout  =  TRUE
         trace  =  FALSE
         maxit  =  750
         size  =  10
         decay  =  0.01
         policy.func  =  "pol3"
```

153 ~ 155

从上面的输出结果可知，该系统应用了交易策略“pol3”，使用了含有 10 个隐藏层且学习的衰变率为 0.01 的神经网络模型。

总之，根据上面的分析结果，如果需要从上面考虑过的模型中选择一个，由于系统“single. nnetR. v12”的不稳定性，我们可以不考虑这个交易系统。然而，在 3.7 节中，我们用剩余的最后 9 年的数据来测试本节中得到这三个最好的交易系统，然后对最好的系统进行最后的评估。

3.7 交易系统

本节描述在最后评估阶段获得的“最佳模型”的结果，这一部分也是模型比较和选择阶段的任务之一。最后评估时期有 9 年的报价数据，我们将在这个时期使用模拟器，应用 5 个选定的交易系统进行交易。

3.7.1 评估最终测试数据

为了把任何一个选定的交易系统应用于最后评估期，我们需要之前紧邻评估期 10 年的报价数据。通过这 10 年的报价数据，我们构建模型，然后应用该模型预测 9 年最终评估期的交易信号。在应用窗口模型的系统中，可能有多个模型介入这些预测。

下面的代码将获取这些系统在 9 年测试期数据上的性能评估指标。

```
> data <- tail(Tdata.train, 2540)
> results <- list()
> for (name in namesBest) {
+     sys <- getVariant(name, fullResults)
+     results[[name]] <- runLearner(sys, Tform, data, Tdata.eval)
+ }
> results <- t(as.data.frame(results))
```

我们对 3 个最优模型进行循环，用最初的训练集数据（10 年）和作为测试集的评估期数据来调用这些模型。这些调用需要应用到之前定义的函数 single( )、函数 slide( ) 和函数 grow( )。之前我们已经看到，这些函数的结果是函数 eval. stats( ) 所给出的一系列性能评价指标。在循环结束后，我们把得到的结果列表转变成一个更方便的表格格式。

下面我们检查一些主要的性能指标值：

```
> results[, c("Ret", "RetOverBH", "MaxDD", "SharpeRatio", "NTrades",
+     "PercProf")]
                    Ret RetOverBH      MaxDD SharpeRatio NTrades PercProf
single.nnetR.v12 -91.13    -61.26 1256121.55       -0.03     759    44.66
slide.nnetR.v15   -6.16     23.71  107188.96       -0.01     132    48.48
grow.nnetR.v12     1.47     31.34   84881.25        0.00      89    53.93
```

从结果中可以确定，在 9 年期间，3 个交易系统中只有其中一个获得了正的回报值，其他系统则都是损失的。在这 3 个系统中，“single. nnetR. v12”系统有很低的回报率，-91.13%，这也确认了该系统的不稳定性。“grow. nnetR. v12”方法看起来明显较好，它不仅有正的回报值，更有一个较小的最大回撤值，其盈利的交易百分比在 50% 以上。但是，在这个测试期，除“single. nnetR. v12”系统外，其他两个系统明显好于简单的买入并持有的市场策略，它们超出买入并持有策略的比例分别为 23.7% 和 31.4%。

最优模型具有以下特点：

```
> getVariant("grow.nnetR.v12", fullResults)

Learner::  "grow"

Parameter values
          learner  =  "nnetR"
          relearn.step  =  120
```

```
linout  =  TRUE
trace  =  FALSE
maxit  =  750
size  =  10
decay  =  0.001
policy.func  =  "pol2"
```

下面我们对最优交易系统在评估期的性能做更深入的分析。为了进行深入的分析，我们需要得到该最优交易系统在评估期的交易记录。函数 grow( ) 不能给出这些记录，所以我们要通过其他方法：

```
> model <- learner("MC.nnetR", list(maxit = 750, linout = T,
+     trace = F, size = 10, decay = 0.001))
> preds <- growingWindowTest(model, Tform, data, Tdata.eval,
+     relearn.step = 120)
> signals <- factor(preds, levels = 1:3, labels = c("s", "h",
+     "b"))
> date <- rownames(Tdata.eval)[1]
> market <- GSPC[paste(date, "/", sep = "")][1:length(signals),
+     ]
> trade.res <- trading.simulator(market, signals, policy.func = "pol2")
```

图 3-9 绘制了这个最优交易系统的交易记录，绘制该图的代码如下：

```
> plot(trade.res, market, theme = "white", name = "SP500 - final test")
```

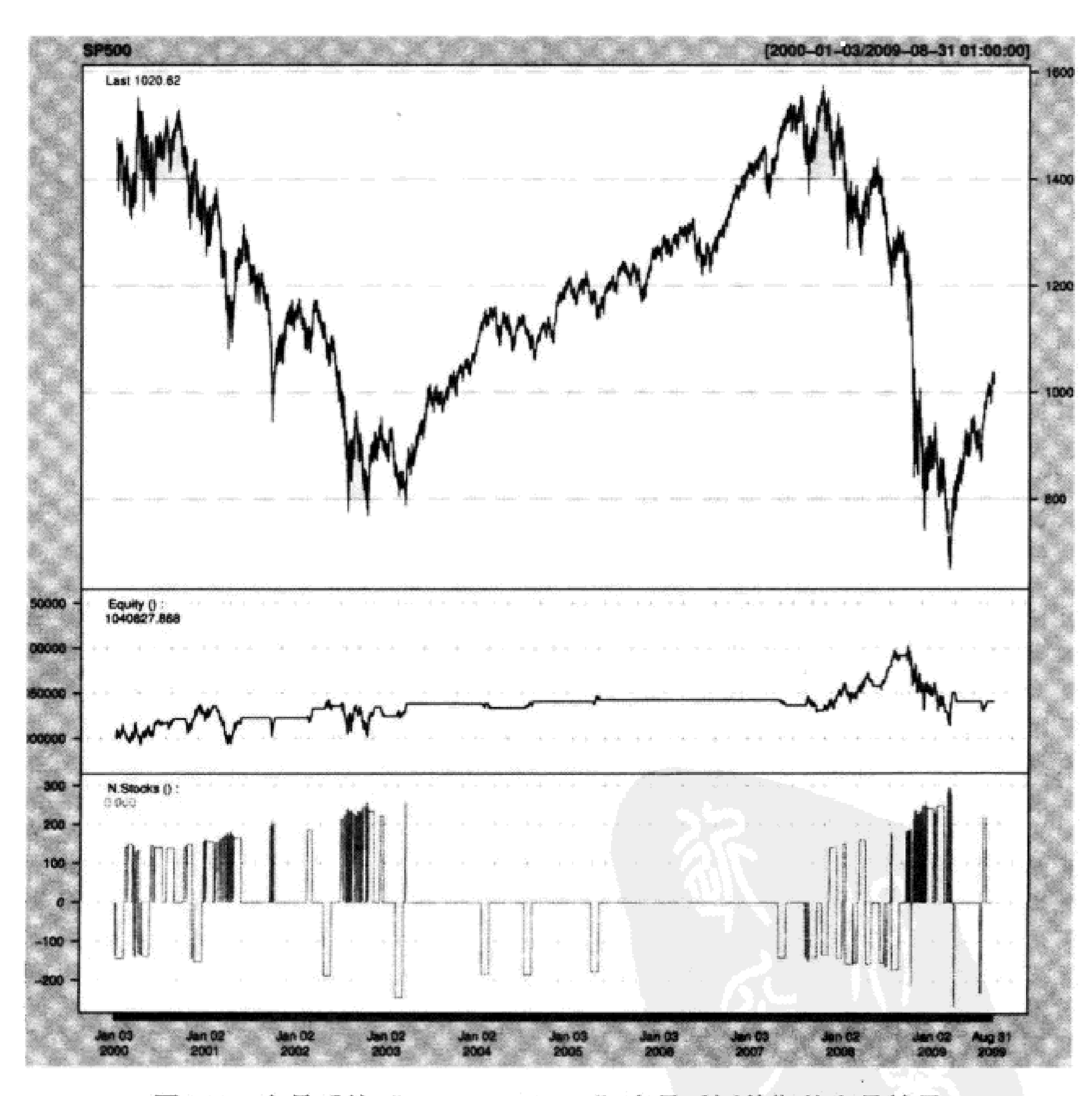


图 3-9　交易系统“grow. nnetR. v12”在最后评估期的交易结果

通过分析图 3-9 可以看出，该系统在很长时间内几乎没有交易活动，即在 2003 年中期到 2007 年中期几乎没有交易行为。令人相当惊讶的是，在这一段时期，市场是显著盈利的。它在某种程度上说明，尽管观察到该系统的整体效果较好，但是它没有表现出它应该具有的性能。另外还注意到，在 2000—2003 年的市场整体下降期间，以及在 2007—2009 年的金融危机中，该交易系统表现得相当好。

性能分析添加包 PerformanceAnalytics 提供了分析任何交易系统性能的大量工具。为更好地理解交易系统的性能，下面我们提供这些工具的一个概览。该添加包的工具可以分析和评估交易策略的回报。可以用如下代码得到策略的回报：

156 ~ 158

```
> library(PerformanceAnalytics)
> rets <- Return.calculate(trade.res@trading$Equity)
```

请注意，函数 Return. calculate( ) 计算的不是我们之前一直应用的收益百分比，它计算出的值乘以因子 100 将等于我们之前应用的收益百分比。

图 3-10 显示策略在所有测试阶段的累积收益率。通过运行下面的代码能够得到图 3-10：

```
> chart.CumReturns(rets, main = "Cumulative returns of the strategy",
+     ylab = "returns")
```

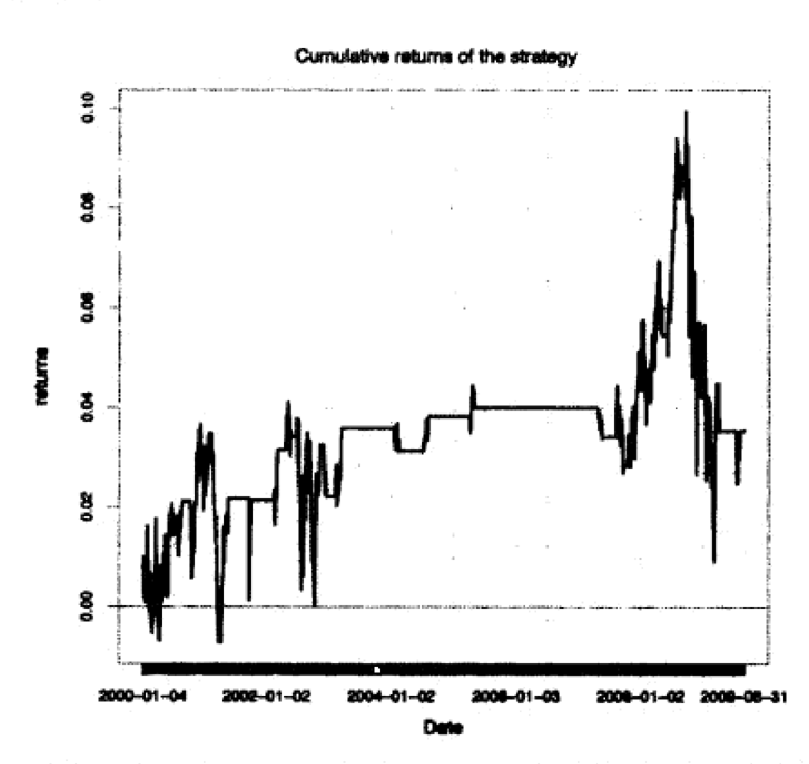


图 3-10　系统“grow. nnetR. v12”在最终评估期的累积收益率

对于大多数的时期，这个系统的收益为正值。在 2008 年中期，系统收益达到最大值 10%。

通常计算年收益率，甚至月收益率也是很有用的。添加包 PerformanceAnalytics 也提供了一些工具来进行这种分析，即函数 yearlyReturn( )：

```
> yearlyReturn(trade.res@trading$Equity)

           yearly.returns
2000-12-29    0.028890251
2001-12-31   -0.005992597
2002-12-31    0.001692791
2003-12-31    0.013515207
2004-12-31    0.002289826
2005-12-30    0.001798355
2006-12-29    0.000000000
```

```
2007-12-31    0.007843569
2008-12-31    0.005444369
2009-08-31   -0.014785914
```

图3-11给出了以上年收益信息的图形，从中可以观察到只有2年的收益为负值。

```
> plot(100*yearlyReturn(trade.res@trading$Equity),
+       main='Yearly percentage returns of the trading system')
> abline(h=0,lty=2)
```

函数 Table. CalendarReturns( ) 给出了更详细的月收益百分比（其中最后一栏是该年的总计）：

```
> table.CalendarReturns(rets)
```

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Equity
2000	-0.5	0.3	0.1	0.2	0	0.2	0.2	0.0	0.0	0.4	0.4	-0.2	1.0
2001	0.0	-0.3	0.2	-0.1	0	0.0	0.0	0.0	0.4	0.0	0.0	0.0	0.3
2002	0.0	-0.1	0.0	-0.2	0	0.0	0.2	0.0	-0.3	-0.1	0.0	0.0	-0.5
2003	0.0	-0.1	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.1
2004	0.1	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2005	0.0	0.0	0.0	-0.2	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.2
2006	0.0	0.0	0.0	0.0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	NA
2007	0.0	0.0	0.0	0.2	0	0.0	0.0	-0.2	0.0	-0.2	0.2	0.1	0.0
2008	-0.3	0.5	0.1	0.1	0	0.0	0.3	0.0	0.9	0.3	0.2	0.3	2.3
2009	-0.5	0.0	-0.2	0.0	0	0.0	0.0	0.0	NA	NA	NA	NA	-0.6

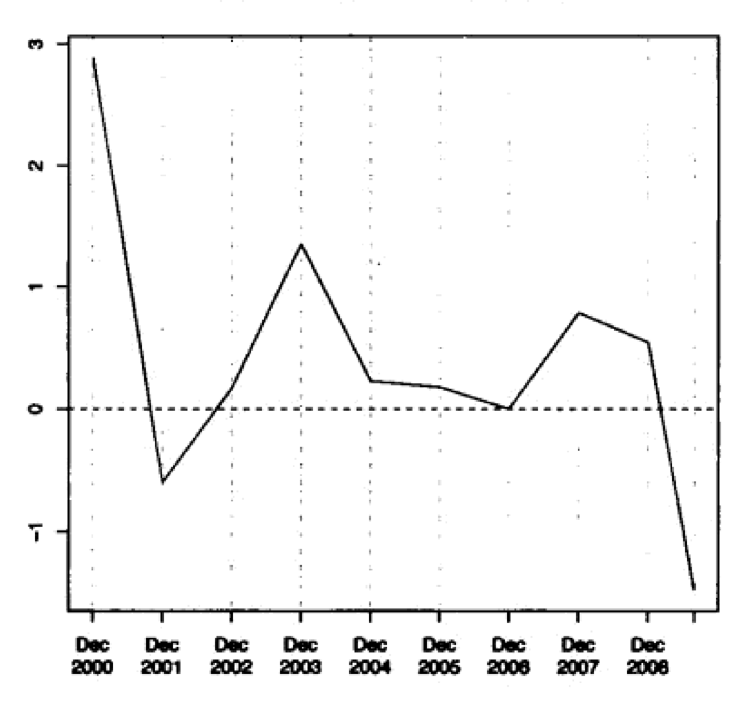


图3-11 "grow. nnetR. v12"系统的年收益信息

上表清楚地显示出系统在很长一段时间内没有交易，许多收益值为0。

最后，我们演示性能分析添加包 PerformanceAnalytics 提供的风险分析工具。这里我们用函数 table. DownsideRisk( ) 来获取交易策略的风险分析信息：

```
> table.DownsideRisk(rets)

                               Equity
Semi Deviation                 0.0017
Gain Deviation                 0.0022
Loss Deviation                 0.0024
Downside Deviation (MAR=210%)  0.0086
Downside Deviation (Rf=0%)     0.0034
```

```
Downside Deviation (0%)          0.0034
Maximum Drawdown                -0.0822
Historical VaR (95%)            -0.0036
Historical ES (95%)             -0.0056
Modified VaR (95%)              -0.0032
Modified ES (95%)               -0.0051
```

该函数给出了多个风险指标信息，其中有百分比最大回撤和半偏差。半偏差目前被认为是一个好于常用的夏普比率的风险度量指标。更多关于这些统计指标的信息可以参阅性能分析添加包 PerformanceAnalytics 的帮助文档。

总的来说，前面所做的分析表明，交易系统 “grow. nnetR. v12” 在 9 年测试期间在较大的风险下有很小的收益。尽管该交易系统的效果明显好于买入并持有策略，这个系统还不能用来管理投资。但是，我们必须说，这个结果是预料之中的。本章的案例是一个相当难的问题，因为有太多的可能性、太多的变体，其中的一些变体已经在本章演示过。本章演示的是所有可能性中的一小部分，如果从这一小部分就能得到一个非常成功的交易系统，那将是很令人吃惊的[⊖]。这不是本案例研究的目的。这里的目标是提供给读者可行的方法，而不是使用这些方法来进行深度的研究而找到最好的交易系统。

159 ~ 161

3.7.2 在线交易系统

假设我们对得到的这个交易系统很满意。我们如何在实际的市场交易中实时应用该系统呢？本节将简略演示一个具有这种实时功能的系统。

这个系统的机制如下。在每天收盘之后，将自动调用该系统。该系统就按照如下步骤操作：1）获得每天最新的可利用数据；2）进行它所需要的建模步骤；3）产生系统调用的结果——即生成一系列的指令。

假设我们将要开发的系统代码保存在文件 “trader. R” 中。每天收盘后调用这个函数的方法由操作系统决定。在基于 UNIX 的系统中，通常有一个名为 “crontab” 的表，你可以把要求操作系统规则性运行的程序加入到这个表中。可以在操作系统的命令行输入下列命令来编辑这个表：

```
shell> crontab -e
```

这个表格中条目的语法相当简单，它由一组描述周期和需要运行命令的字段构成。在下面的例子中，它指示系统在每个工作日的下午 7 点运行程序 “trader. R”，具体代码如下：

```
0 19 * * 1-5 /usr/bin/R --vanilla --quiet < /home/xpto/trader.R
```

开始两项分别代表分钟和小时。第三项和第四项分别代表所在月份的哪一天、月份，而星号（“ * ”）意味着对该域的所有情况都运行本程序。第五项代表星期几，1 代表星期一，用 “ - ” 来表明日期区间。最后一项是需要运行的程序。

程序 “trader. R” 实现的一般算法如下：

- 读入当前交易系统的状态。
- 读入所有可以获得的最新数据。
- 检查是否需要重新训练模型。
- 获得今天的预测信号。
- 用该预测信号来调用策略函数，并获得交易指令。
- 输出今天的交易指令。

交易系统的当前状态是一个数据结构，它存储交易系统在日常运行中要求记忆的信息。在

⊖ 如果我们公开这样一个相当成功的系统也是同样令人吃惊的！

我们的案例中，它应该包括目前的 NNET 模型、用来得到该模型的参数、用来获取模型的训练数据和相关的模型规范、模型的“年龄”（它对于何时需要重新训练模型很重要）、到今天为止系统的交易记录信息和现在的持仓头寸。理想的情况下，这些信息应该在一个数据库中，然后交易系统应用 R 的数据库界面来访问这些信息（参见 3.2.4 节）。注意，有关持有头寸的信息需要根据交易系统外部的信息来更新，因为决定开仓和平仓的时机是市场。而我们的交易系统则相反，它假设所有的交易在下一天的开始完成。 162

如果我们设置好数据模型，那么获取新的数据就很容易。在 3.3.2 节中提到，可以用函数 getModelData( ) 获取最近的报价，从而更新数据集。

模型参数 relearn. step 和模型记忆的所有其他参数一样，它设置何时需要重新训练模型。如果模型的“年龄”大于参数 relearn. step 的值，那么该模型需要重新训练。如果模型需要重新训练，那么我们应该用当前窗口的数据调用函数 MC. nnetR( )，从而获得新的模型。由于我们的最佳交易系统使用的是增长窗口模式，所以训练集数据不断增长。如果它变得太大而超出计算机的内存，这可能就成为一个问题。如果这种情况发生，可以考虑丢弃太旧的训练集数据，从而修剪训练集数据使得它的大小可以被计算机接受。

最后，我们必须得到今天的交易信号预测。这意味着需要用当前的预测模型调用函数 predict( ) 以获得训练集最后一行的预测值，即今天的预测值。有了这个预测值，就可以调用具有适当参数的交易策略，从而得到今天的交易指令集合。这就是最终的程序结果。

上面的大概轮廓给你提供了实现一个实时交易系统的足够信息。

3.8　小结

本章的主要目的是向读者介绍一个更加实用的数据挖掘案例。本章描述的案例具有多个挑战性：1）处理时间序列数据；2）处理可能具有模式变化的非常动态的系统；3）从预测模型转到应用领域的具体行动。

从方法术语的角度，本章介绍了下列新的主题：

- 时间序列建模。
- 处理带有窗口机制的模式变化。
- 人工神经网络。
- 支持向量机。 163
- 多元自适应回归样条。
- 用蒙特卡罗方法来评估时间序列模型。
- 几个预测稀有事件或者金融交易系统性能的新的评价指标。

从学习 R 的角度，我们演示了：

- 如何处理时间序列。
- 如何从不同的来源读入数据，例如数据库。
- 如何获得多种新的模型（支持向量机、神经网络、多元自适应回归样条）。
- 如何应用多个用于金融模型的 R 添加包。 164

侦测欺诈交易

数据挖掘的一个常见问题是侦测一个现象中的特殊观测值，第三个案例就是这种问题的一个实际案例。也就是说，找到罕见的和特殊的观测值。这类应用的动因是某些公司的销售员所报告的一系列产品的交易情况。目的是找到“奇怪的”交易记录报告，它可能指出某些销售员涉嫌欺诈。数据挖掘的结果有助于公司的事后检查活动。由于公司分配给事后检查工作的资源有限，所以希望数据挖掘过程能提供某种欺诈概率排序作为输出结果。这些排序可以使公司以最佳方式来利用其事后检查资源。这种总资源有限的检查活动在很多领域有广泛应用，如信用卡交易、税务申报检验等。本章讨论多个新的数据挖掘主题：1）离群值或异常值检验；2）聚类分析；3）半监督预测模型。

4.1 问题描述与目标

考虑到在经济和社会领域中经常存在欺诈交易等非法活动，因此欺诈检验是数据挖掘技术的一个重要应用领域。从数据分析的角度，欺诈行为通常与异常的观测值相关联，因为这些欺诈行为是偏离常规的。在多个数据分析领域，这些偏离常规的行为经常称为离群值。事实上，离群值的标准定义是“一个观测值与其他观测值偏离太多而引起猜疑，认为它产生于不同的机制”(Hawkins, 1980)。

本案例使用的数据是某公司的销售员所报告的交易数据。这些销售员负责销售该公司的产品并定期报告销售情况。我们得到的数据是一个较短时期内的销售数据。销售员可按照自己的策略和市场情况来自由设置销售价格。每月末，他们向公司报告销售情况。数据挖掘应用的目的是根据公司过去发现的交易报告中的错误和欺诈企图，帮助公司完成核实这些销售报告真实性的工作。我们提供一份欺诈率排名的报告，这个欺诈率排名将允许公司把有限的检验资源分配给系统所提示的更“可疑”的那些报告。

165

4.2 可用的数据

我们所用的数据来自一个未公开的渠道并被匿名。数据共计401 146行，每一行包括来自销售员报告的信息。这些信息包括销售员的ID号、产品编号、销售员所报告的销售数量和总价值。该公司已经对这些数据进行过一些分析，分析的结果显示在最后一列，它们是公司对交易进行检查的结果。总之，我们应用的数据集包括以下各列：

- ID：说明销售员ID的一个因子变量。
- Prod：说明销售产品ID号的一个因子变量。
- Quant：报告该产品销售的数量。
- Val：报告销售记录的总价值。
- Insp：有3个可能值的因子变量——ok表示公司检查了该交易并认为该交易有效；fraud表示发现该交易为欺诈；unkn表示该交易未经过公司审查。

4.2.1 加载数据至 R

该数据集可在本书的 R 添加包或者本书网站上获得。在本书网站上，本案例的数据是一个 Rdata 文件，它是包含本案例数据集的一个数据框。如果要使用本书网站的数据，需要先下载这个文件到计算机的本地目录下，然后运行命令：

```
> load("sales.Rdata")
```

如果当前运行 R 的目录是下载文件的目录，那么该命令将从这个文件中把数据载入到一个名为 sales 的数据框。

如果使用本书添加包中的数据，那么应该进行如下操作：

166

```
> library(DMwR)
> data(sales)
```

而且，得到一个名为 sales 的数据框，可以用下面的代码显示其前几行数据，如下所示：

```
> head(sales)

  ID Prod Quant   Val Insp
1 v1   p1   182  1665 unkn
2 v2   p1  3072  8780 unkn
3 v3   p1 20393 76990 unkn
4 v4   p1   112  1100 unkn
5 v3   p1  6164 20260 unkn
6 v5   p2   104  1155 unkn
```

4.2.2 探索数据集

为了初步了解数据的统计特征，可以使用函数 summary() [⊖]，代码如下：

```
> summary(sales)

       ID                Prod               Quant                  Val
 v431   : 10159   p1125   :   3923   Min.   :      100   Min.   :    1005
 v54    :  6017   p3774   :   1824   1st Qu.:      107   1st Qu.:    1345
 v426   :  3902   p1437   :   1720   Median :      168   Median :    2675
 v1679  :  3016   p1917   :   1702   Mean   :     8442   Mean   :   14617
 v1085  :  3001   p4089   :   1598   3rd Qu.:      738   3rd Qu.:    8680
 v1183  :  2642   p2742   :   1519   Max.   :473883883   Max.   :4642955
 (Other):372409   (Other):388860     NA's   :    13842   NA's   :    1182
    Insp
 ok   : 14462
 unkn :385414
 fraud:  1270
```

从结果可知，数据集中有大量的产品和销售人员信息，可以使用函数 nlevels() 来确认这一点：

```
> nlevels(sales$ID)

[1] 6016

> nlevels(sales$Prod)

[1] 4548
```

167

函数 summary() 的结果揭示了数据集的几个事实。首先，在 Quant 列和 Val 列有大量的缺失

⊖ 可以用 R 添加包 Hmisc 中的函数 describe () 来得到类似的有趣结果，留给读者自己练习。

值。如果在同一个交易中这两者缺失，就会产生比较严重的问题，因为这将导致一条销售交易中有关销售量的关键信息缺失。可以很容易地检查是否有这种情况存在，代码如下：

```
> length(which(is.na(sales$Quant) & is.na(sales$Val)))

[1] 888
```

可以看到，具有以上问题交易的数量是合理的。鉴于总的交易数量很大，人们可以质疑是否可以简单地删除这些同时缺失 Val 和 Quant 列的报告。我们将在 4.2.3 节中考虑这个问题和其他的方法。

附带一提，特别是对特大数据集，有更高效的方式获取以上信息。在上面的命令中，虽然看起来使用函数 length( ) 和函数 which( ) 更容易理解，但可以利用 R 中的逻辑值表达式（T = 1，F = 0）更高效地得到上面的结果：

```
> sum(is.na(sales$Quant) & is.na(sales$Val))

[1] 888
```

从函数 summary( ) 的结果得到的另一个有趣的结论是检查列的数据分布。实际上，正如预期的那样，欺诈行为的比例相对较低。即使只考虑那些已经检查过的销售记录，欺诈行为的比例总体而言也是较低的：

```
> table(sales$Insp)/nrow(sales) * 100

        ok       unkn      fraud
 3.6051712 96.0782359  0.3165930
```

图 4-1 显示了每个销售人员报告的数量。可以确认的是，所有销售人员的数据相当不同。图 4-2显示了类似的上述数据，但是是针对每个产品的。我们再一次看到了很大的变动性，上述两个图形可以用下面代码得到：

168

```
> totS <- table(sales$ID)
> totP <- table(sales$Prod)
> barplot(totS, main = "Transactions per salespeople", names.arg = "",
+     xlab = "Salespeople", ylab = "Amount")
> barplot(totP, main = "Transactions per product", names.arg = "",
+     xlab = "Products", ylab = "Amount")
```

图 4-1　每个销售人员的交易数量

变量 Quant 和 Val 的描述性统计量显示了相当明显的变动性。这表明报告中的产品可能有很大的不同，因此对不同产品分别进行处理是有意义的。实际上，如果两个交易报告的产品是相同的，而产品的标准价格差别太大，那么其中的一个交易报告只能视为不正常。不过，用这两个数量得出这样的结论可能不是太理想的根据。实际上，由于每个交易中销售的产品数量不同，所以用单位产品的价格来进行上面的分析可能更正确。可以把这个单位产品价格作为新的一列加入到数据框中，代码如下：

```
> sales$Uprice <- sales$Val/sales$Quant
```

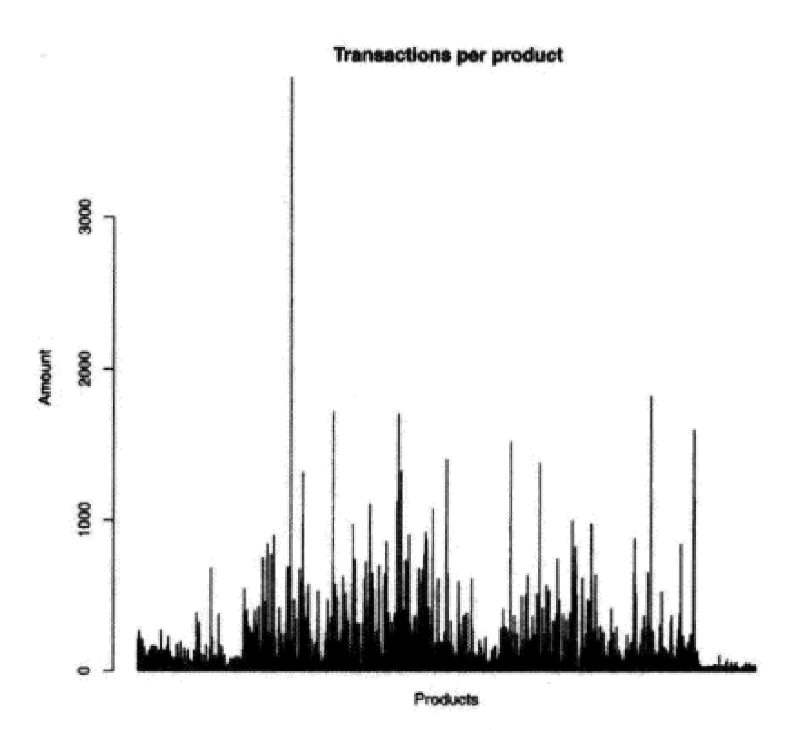


图 4-2　每个产品的交易数量

169

在交易中，同一产品的单位交易价格应该是相对稳定的。当分析一个较短时期内的交易时，产品的单位价格不应该出现巨大变化。

我们可以用如下代码来检查产品单位价格的分布：

```
> summary(sales$Uprice)

      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
2.4480e-06 8.4600e+00 1.1890e+01 2.0300e+01 1.9110e+01 2.6460e+04
      NA's
1.4136e+04
```

我们再次看到明显的变动性。

鉴于这些事实，我们不可避免地要分别对每个产品的交易进行分析，据此找出每个产品的可疑交易。应用这种方法的一个问题是部分产品只有非常少的交易。实际上，在 4548 种产品中，有 982 种产品的交易量小于 20。基于这种小于 20 的样本得出一个报告为不寻常报告有较大的风险。

检查最贵的和最便宜的产品可能是很有趣的。我们用单位价格的中位数来代表已售产品的标准价格。下面代码用于获取我们所需要的信息：

170

```
> attach(sales)
> upp <- aggregate(Uprice,list(Prod),median,na.rm=T)
> topP <- sapply(c(T,F),function(o)
+                upp[order(upp[,2],decreasing=o)[1:5],1])
> colnames(topP) <- c('Expensive','Cheap')
> topP
```

```
     Expensive Cheap
[1,] "p3689"   "p560"
[2,] "p2453"   "p559"
[3,] "p2452"   "p4195"
[4,] "p2456"   "p601"
[5,] "p2459"   "p563"
```

我们把数据框添加到 R 中，这样便于直接应用数据框的列名。我们使用 aggregate( ) 函数得到每个产品的单位价格的中位数。函数 aggreagte( ) 把某个产生数值的函数（聚合函数）应用在按照某些因子（或因子列表）形成的数据子组上。其结果是一个数据框，数据框的值为每个组的聚合函数值。从上面获得的数据框，我们通过应用函数 sapply( ) 并更改函数 order( ) 的参数 decreasing 的值，得到 5 个最昂贵（最便宜）的产品。

我们可以用这 5 个产品的单位价格的箱图来确认它们完全不同的价格分布：

```
> tops <- sales[Prod %in% topP[1, ], c("Prod", "Uprice")]
> tops$Prod <- factor(tops$Prod)
> boxplot(Uprice ~ Prod, data = tops, ylab = "Uprice", log = "y")
```

运算符 "%in%" 用于测试一个值是否属于一个集合。上面代码调用函数 factor( ) 十分必要，否则数据框 tops 的 Prod 列的水平数将和原始的 sales 数据框的 Prod 列的水平数相同。这将导致 boxplot( ) 函数为每个水平值绘制一个箱图。最昂贵价格和最便宜价格的数据量纲是相当不同的。为了避免最便宜产品的数据变得无关紧要，我们在图形中取了对数。这通过设置参数 log = y 来实现，它表明 Y 轴是原始数据的对数（注意，数轴上的同一距离所对应的单位价格的不同范围）。上述代码的结果如图 4-3 所示。

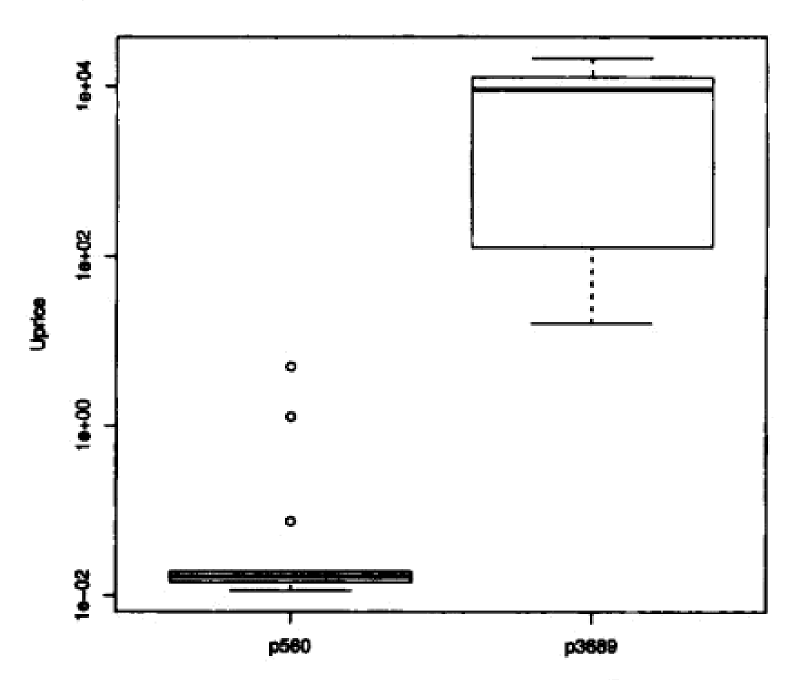


图 4-3　最便宜和最昂贵产品的单位价格分布

可以进行类似的分析，以找出那些给公司带来更多（少）资金的销售人员：

```
> vs <- aggregate(Val,list(ID),sum,na.rm=T)
> scoresSs <- sapply(c(T,F),function(o)
+                    vs[order(vs$x,decreasing=o)[1:5],1])
> colnames(scoresSs) <- c('Most','Least')
> scoresSs

     Most    Least
[1,] "v431"  "v3355"
[2,] "v54"   "v6069"
```

```
[3,] "v19"   "v5876"
[4,] "v4520" "v6058"
[5,] "v955"  "v4515"
```

有趣的是，上面列表中给公司带来更多资金的前100名销售人员的资金收入几乎占公司资金收入的40%，而在6016名销售人员中，底部2000人的总收入不足公司总收入的2%。这可能给出某些提示，说明该公司需要进行内部改革。

```
> sum(vs[order(vs$x, decreasing = T)[1:100], 2])/sum(Val, na.rm = T) *
+     100

[1] 38.33277

> sum(vs[order(vs$x, decreasing = F)[1:2000], 2])/sum(Val,
+     na.rm = T) * 100

[1] 1.988716
```

如果我们对每个产品所销售的数量进行类似的分析，结果更加不平衡：

```
> qs <- aggregate(Quant,list(Prod),sum,na.rm=T)
> scoresPs <- sapply(c(T,F),function(o)
+                    qs[order(qs$x,decreasing=o)[1:5],1])
> colnames(scoresPs) <- c('Most','Least')
> scoresPs

     Most    Least
[1,] "p2516" "p2442"
[2,] "p3599" "p2443"
[3,] "p314"  "p1653"
[4,] "p569"  "p4101"
[5,] "p319"  "p3678"
> sum(as.double(qs[order(qs$x,decreasing=T)[1:100],2]))/
+   sum(as.double(Quant),na.rm=T)*100

[1] 74.63478

> sum(as.double(qs[order(qs$x,decreasing=F)[1:4000],2]))/
+   sum(as.double(Quant),na.rm=T)*100

[1] 8.94468
```

你可能已经注意到上面的代码使用了 as.double( ) 函数。这种情况下，sum( ) 函数产生的数目太大，必须用 double 类型来存储。因此，应用函数 as.double( ) 能确保这种类型转换。

在4548个产品中，其中4000个产品代表了少于10%的销量，而销量最高的100个产品占了近75%的销量。这个信息对产品的生产十分有用。特别地，它并不意味着公司应该停止生产销售量较少的产品。事实上，如果这些产品有更高的边际利润，就会更有利可图。由于我们没有关于产品制造成本的任何信息，所以无法得出是否继续生产销量小的产品的任何结论。

我们为了发现异常交易报告，做出的主要假设之一是，所有产品的单位价格都遵循一个接近正态的分布。这意味着，我们预期的同一产品的单位价格大致相同，它们之间小的变化可能是由于销售人员为了达到他们的商业目标而采取的策略。在这种假设下，有一些基本的统计检验技术来帮助我们发现价格是否偏离正态假设。其中一个方法是基于箱图规则。在本书中，我们已经多次看到用箱图来侦测离群值。箱图规则的定义是：如果一个观测值高于上须（或者低于下须），则将观测值标记为异常高（低）。上须的定义是 $Q_3 + 1.5 \times \text{IQR}$（下须的定义为 $Q_1 - 1.5 \times \text{IQR}$），其中 Q_1 是第一个四分位数，Q_3 是第三个四分位数，IQR 是四分位距，其定义为 $\text{IQR} = (Q_3 - Q_1)$。这个简单的规则对正态分布的变量很有效。由于它是基于四分位数这一稳健的统计

量，所以在有少数离群值时该规则是稳健的。下面的代码可以（根据上述定义）确定每个产品的异常值个数：

```
> out <- tapply(Uprice,list(Prod=Prod),
+               function(x) length(boxplot.stats(x)$out))
```

函数 boxplot. stats( ) 可以获得某些用于绘制箱图的统计量。它返回含有这些信息的一个列表。列表的 out 元素包含了根据箱图规则被视为离群值的观测值。上面的代码计算了每个产品交易中含有的离群值数量。含有较多离群值的产品如下所示：

171～173

```
> out[order(out, decreasing = T)[1:10]]

Prod
p1125 p1437 p2273 p1917 p1918 p4089  p538 p3774 p2742 p3338
  376   181   165   156   156   137   129   125   120   117
```

应用这个简单方法，找到 29 446 个被认为是离群值的交易，这相当于总交易数量的 7%。

```
> sum(out)

[1] 29446

> sum(out)/nrow(sales) * 100

[1] 7.34047
```

也许读者会怀疑这个简单的确定离群值的规则是否能充分提供这个案例中所需要的帮助。4.4.1.1 节将对这个规则稍加修改以适应我们的应用，并评估其性能。

需要注意本节中得出的某些结论。我们使用的数据独立于下面的事实：有些报告发现有欺诈，而其他的报告可能也有欺诈，只是没有探测到。这意味着有些错误数据可能导致我们的某些结论是有偏差的。这里的问题是，对那些标记为有欺诈的报告，我们不知道真的是否有欺诈。理论上，我们肯定正确的交易是那些在列 Insp 取值为 OK 的报告，然而这些报告只占总交易量的 3.6%。所以，尽管分析是正确的，但结果可能受低质量的数据影响。这在现实问题中应该考虑到，不要基于包含错误的数据来给公司提供建议。然而完整检查数据是不可能的，因此这种风险总是会存在的。至多我们在探索性分析数据时避免使用已经发现包含错误的较少的交易数据。另一个可以做的事情就是把结果呈现给公司，如果他们认为某些分析结果是非期望的，可以对这些导致惊奇结果的数据进行更细致的分析。这意味着这种分析通常需要与领域专家以某种形式进行沟通，特别是像本案例中这样怀疑数据的质量时。此外，这种类型的探索性分析对低质量的数据尤其重要，因为许多问题可以容易地在这个阶段发现。

174

4.2.3 数据问题

数据质量问题可能会给应用本章后面技术带来一些障碍，本节将介绍一些数据质量问题。

4.2.3.1 缺失值

我们从处理缺失值这一问题开始。在 2.5 节中提到，基本上有 3 个选择：1）剔除这些个案；2）用某些策略来填补缺失数据；3）运用可以处理缺失值的工具。考虑到本章将要用到的工具，只有前两个选择对我们是有效的。

前面提到过，主要的问题是变量 Quant 和变量 Val 都有数据缺失的交易。如果移除所有的 888 个个案将导致剔除某些产品或销售人员的大部分交易，这时候全部剔除 888 个个案是有问题的。下面，我们检查这种情况是否会发生。

每个销售人员和产品的交易数量由下面的代码给出：

```
> totS <- table(ID)
> totP <- table(Prod)
```



可以用下面的代码显示与存在问题的交易相关的销售人员和产品：

```
> nas <- sales[which(is.na(Quant) & is.na(Val)), c("ID", "Prod")]
```

下面给出变量 Quant 和变量 Val 同时有缺失值的交易占很大比例的销售人员：

```
> propS <- 100 * table(nas$ID)/totS
> propS[order(propS, decreasing = T)[1:10]]

    v1237     v4254     v4038     v5248     v3666     v4433     v4170
13.793103  9.523810  8.333333  8.333333  6.666667  6.250000  5.555556
    v4926     v4664     v4642
 5.555556  5.494505  4.761905
```

至少从销售人员方面来看，直接剔除这些同时在两个变量有缺失值的交易是合理的，因为它们只代表了很小的一部分交易。另外，试图同时填补那两列似乎更为冒险。

从产品方面来看，应用下面的代码得到同时在两个变量都有缺失值的交易占较大的比例的产品：

```
> propP <- 100 * table(nas$Prod)/totP
> propP[order(propP, decreasing = T)[1:10]]

   p2689    p2675    p4061    p2780    p4351    p2686    p2707    p2690
39.28571 35.41667 25.00000 22.72727 18.18182 16.66667 14.28571 14.08451
   p2691    p2670
12.90323 12.76596
```

多个产品将被删除的交易超过 20%；特别是产品 p2689 将有近 40% 的交易被删除。这看起来太多了。另一方面，如果我们决定填补那些缺失值，那么唯一合理的策略是使用相同产品的具有“完全”信息的交易。这意味着运用该产品剩余 60% 的交易信息去填补该产品 40% 的交易信息。这看起来也是不合理的。幸运的是，如果我们看看产品的单位价格分布之间的相似性（详见 4.2.3.2 节），那么我们实际上可以观察到这些产品和其他产品相当类似。这种情况下，我们可以得到如下结论：如果剔除有缺失数据的交易后只有少量的交易，那么我们就可以和类似产品的交易相结合以增加离群值检测的统计可靠性。总之，剔除所有同时在数量和价格上有缺失值的交易是最好的选择： 175

```
> detach(sales)
> sales <- sales[-which(is.na(sales$Quant) & is.na(sales$Val)),]
```

我们用 detach( ) 函数来禁止直接访问数据框的列。原因是基于 attach( ) 函数的工作方式。当调用函数 attach（sales）时，R 复制 sales 数据框的每一列，为每一列建立一个新对象。如果从 sales 数据框删除数据，那么这些变化不会反映在这些新对象中。总的来说，当要查询的数据容易发生改变时，不应该使用 attach( ) 函数提供的便利；否则，我们可能会得到不一致的数据视图：原始数据框的数据视图和函数 attach( ) 所创建对象的视图。后者是在一定时间内的原始数据框的“快照”，如果在调用函数 attach( ) 后我们修改了数据框，那么这个“快照”就过时了。

现在让我们分析剩余的在数量或者价格变量上有缺失值的交易。我们从计算每一种产品在数量上有缺失值的交易比开始：

```
> nnasQp <- tapply(sales$Quant,list(sales$Prod),
+                  function(x) sum(is.na(x)))
> propNAsQp <- nnasQp/table(sales$Prod)
> propNAsQp[order(propNAsQp,decreasing=T)[1:10]]

    p2442     p2443     p1653     p4101     p4243      p903     p3678
1.0000000 1.0000000 0.9090909 0.8571429 0.6842105 0.6666667 0.6666667
    p3955     p4464     p1261
0.6428571 0.6363636 0.6333333
```

p2442 和 p2443 两个产品所有的交易数量是缺失的。因为我们无法计算其标准单价，所以没有进一步的信息，用这些产品的交易信息不可能进行任何分析。它们一共有 54 份报告，其中两份标记为欺诈，而其他的标记为“OK”。这意味着，或者检查员掌握了比这个数据集更多的信息，或者我们得到的数据有输入错误，因为从这些交易中似乎不可能得到任何结论。基于此，我们将删除这些交易报告：

176

```
> sales <- sales[!sales$Prod %in% c("p2442", "p2443"), ]
```

由于我们从数据集中删除了两种产品，所以我们应该更新列 Prod 的水平：

```
> nlevels(sales$Prod)

[1] 4548

> sales$Prod <- factor(sales$Prod)
> nlevels(sales$Prod)

[1] 4546
```

是否有销售人员的所有交易数量为缺失值？

```
> nnasQs <- tapply(sales$Quant, list(sales$ID), function(x) sum(is.na(x)))
> propNAsQs <- nnasQs/table(sales$ID)
> propNAsQs[order(propNAsQs, decreasing = T)[1:10]]

    v2925     v5537     v5836     v6058     v6065     v4368     v2923
1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 0.8888889 0.8750000
    v2970     v4910     v4542
0.8571429 0.8333333 0.8095238
```

从上面结果可以看到，有几个销售人员没有在报告中填写交易的数量信息。然而，这种情况下，问题没有那么严重。实际上，应用单位价格应该类似的假设，只要我们有其他销售人员报告的相同产品的交易，我们就可以尝试使用此信息来填补那些缺失值。正因为如此，我们将不会删除这些交易。

现在我们将对交易的 Val 列的缺失值进行类似的分析。首先，每种产品在该列有缺失值的交易所占的比例为：

177

```
> nnasVp <- tapply(sales$Val,list(sales$Prod),
+                  function(x) sum(is.na(x)))
> propNAsVp <- nnasVp/table(sales$Prod)
> propNAsVp[order(propNAsVp,decreasing=T)[1:10]]

     p1110      p1022      p4491      p1462        p80      p4307
0.25000000 0.17647059 0.10000000 0.07500000 0.06250000 0.05882353
     p4471      p2821      p1017      p4287
0.05882353 0.05389222 0.05263158 0.05263158
```

这些数字是合理的，因此删除这些交易是没有意义的。我们将用其他的交易来填补这些缺失值。从销售人员方面，计算这些数字的方法如下：

```
> nnasVs <- tapply(sales$Val, list(sales$ID), function(x) sum(is.na(x)))
> propNAsVs <- nnasVs/table(sales$ID)
> propNAsVs[order(propNAsVs, decreasing = T)[1:10]]

     v5647        v74      v5946      v5290      v4472      v4022
0.37500000 0.22222222 0.20000000 0.15384615 0.12500000 0.09756098
      v975      v2814      v2892      v3739
0.09574468 0.09090909 0.09090909 0.08333333
```

同样，这个比例不是太高。

在这个阶段中，我们已经删除了所有没有足够信息来填补缺失值的报告。对剩余的缺失值，基于同一个产品的交易单位价格相似的假设，我们将应用某个方法进行缺失值填补。我们将首先获得每一个产品的标准单价。计算标准价格时，我们将跳过标记为欺诈的交易价格。对剩余的交易，我们将使用每个产品单位价格的中位数作为相应产品的标准价格：

```
> tPrice <- tapply(sales[sales$Insp != "fraud", "Uprice"],
+     list(sales[sales$Insp != "fraud", "Prod"]), median, na.rm = T)
```

因为我们目前没有交易同时在这两个变量上有缺失值，所以每一个产品有了一个标准单价后，我们就可以用它来计算两个可能的缺失值（Quant 和 Val）。下面的代码将填补所有剩余的缺失值：

```
> noQuant <- which(is.na(sales$Quant))
> sales[noQuant,'Quant'] <- ceiling(sales[noQuant,'Val'] /
+                                 tPrice[sales[noQuant,'Prod']])
> noVal <- which(is.na(sales$Val))
> sales[noVal,'Val'] <- sales[noVal,'Quant'] *
+                        tPrice[sales[noVal,'Prod']]
```

我们填补了12 900个未知的数量值，并填补了294个交易总价值。如果你像我一样，我相信你也会欣赏上面这些完成所有操作的精简代码。它用的都是索引方法！我们使用 ceiling() 函数来避免 Quant 的非整数值。这个函数返回不小于参数中数值的最小整数。

鉴于我们现在有了 Quant 和 Val 的所有值，我们就可以重新计算 Uprice 列的值来填充先前未知的单位价格：

```
> sales$Uprice <- sales$Val/sales$Quant
```

178

经过所有这些预处理步骤后，有一个没有任何缺失值的数据集。为了进一步的分析，保存当前的 sales 数据框，这样可以从这个数据框来进行下面的分析，而不必再重复前面的所有步骤。可以如下保存数据集：

```
> save(sales, file = "salesClean.Rdata")
```

函数 save() 可以把任何对象保存在参数 file 指定的文件中。在4.2.1节中提到过，这些文件保存的对象可以通过 load() 函数载入到 R 中。

4.2.3.2 只有少量交易的产品

有些产品只有极少的交易。因为需要用这些交易的信息来确定交易中是否有异常，所以这是一个问题。如果有太少的交易，在要求的统计显著性下很难做出决定。这种情况下，考虑是否可以和一些产品的交易一起分析来避免这个问题。

尽管完全缺乏产品之间关系的信息，但可以尝试通过观察产品单价分布之间的相似性来推断其中的一些关系。如果我们发现具有类似价格的产品，那么我们可以考虑合并它们相应的交易并对它们一起进行分析，从而寻找异常值。比较两个分布的一种方法是可视化检查法。鉴于产品的数量，这是不可行的。另一种方法是比较总结分布的一些统计特性。连续变量分布的两个重要属性是集中趋势和离散指标。如前所述，假设任何产品的单位价格分布大约为正态是合理的。这意味着虽然价格存在变化性，但它们应该很好地分布在常见价格的周围。但是，这里也假设有离群值的出现，它们可能是由于欺诈企图或错误造成的。这里使用中位数作为衡量中心的统计量，应用四分位距（IQR）作为离散指标的统计量更有意义。与更常用的均值和标准差相比，这些统计量在有离群值存在时更加稳健。计算每种产品所有交易的这两个统计量的代码如下：

```
> attach(sales)
> notF <- which(Insp != 'fraud')
> ms <- tapply(Uprice[notF],list(Prod=Prod[notF]),function(x) {
+       bp <- boxplot.stats(x)$stats
+       c(median=bp[3],iqr=bp[4]-bp[2])
+     })
> ms <- matrix(unlist(ms),
+              length(ms),2,
+              byrow=T,dimnames=list(names(ms),c('median','iqr')))
> head(ms)

      median      iqr
p1 11.346154 8.575599
p2 10.877863 5.609731
p3 10.000000 4.809092
p4  9.911243 5.998530
p5 10.957447 7.136601
p6 13.223684 6.685185
```

上面代码使用函数 boxplot. stats( ) 获得中位数、第一个四分位数和第三个四分位数。对每个产品的所有交易，计算这些统计量，从分析中剔除有欺诈的交易。有了这些统计量后，得到含有每个产品的中位数和四分位距的一个矩阵。

图 4-4a 是根据每个产品的中位数和 IQR（四分位距）绘制的图形。因为有几个产品在这些统计量上有很大的值，所以图 4-4 很难阅读。特别是，产品 p3689（在右上角的点）明显地与该公司的其他产品不同。通过使用对数尺度来克服这个可视化的问题（图 4-4b）。在图 4-4b 中，使用黑色的“+”标志，表明该产品的交易数少于 20。绘制图 4-4 的代码如下，其中参数 log = *xy* 用来设置图形的两个数轴的标度为对数标度：

```
> par(mfrow = c(1, 2))
> plot(ms[, 1], ms[, 2], xlab = "Median", ylab = "IQR", main = "")
> plot(ms[, 1], ms[, 2], xlab = "Median", ylab = "IQR", main = "",
+      col = "grey", log = "xy")
> smalls <- which(table(Prod) < 20)
> points(log(ms[smalls, 1]), log(ms[smalls, 2]), pch = "+")
```

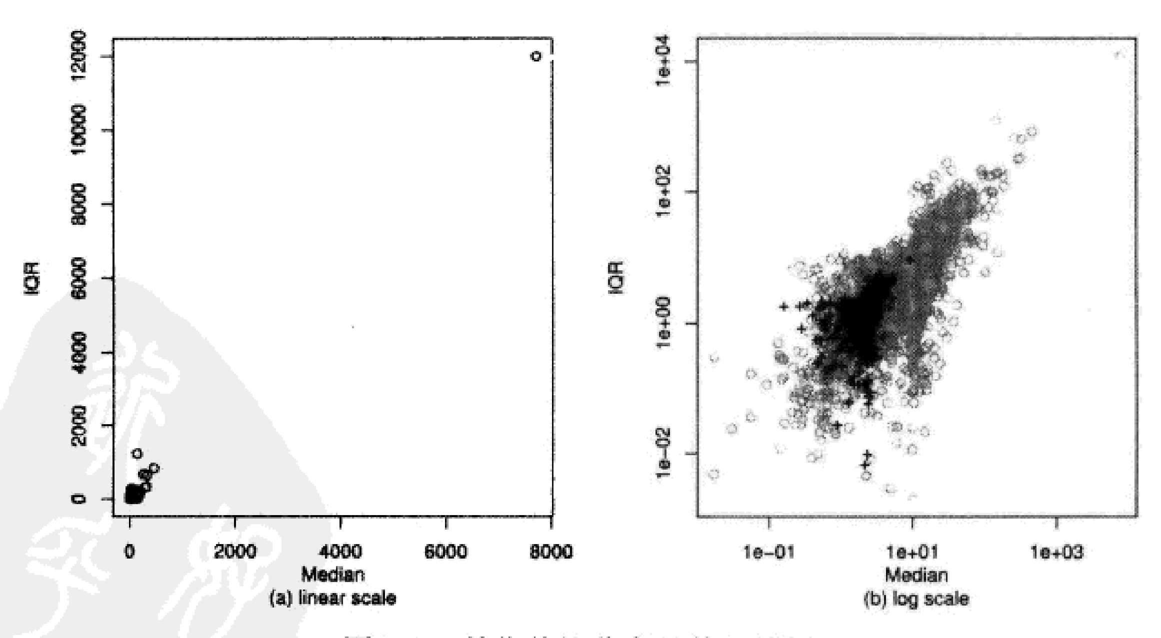


图 4-4　单位价格分布的某些特征

在图 4-4b 中，第一个注意到的是，许多产品的中位数和 IQR 大致相同，即使是对数标度。这提供了单价分布相似性的一个很好证据。此外，我们可以看到，那些有少数交易的产品中有很

多和其他产品非常类似。然而，也有几种产品，它们不仅有很少的交易，它们的单位价格分布也相当不同。显然我们很难判断这些产品是否为欺诈交易。

尽管可视化检查单位价格的分布特征有以上的优点，但是当比较两种产品的分布时，还需要有正式的检验以获取更高的精确度。因为非参数检验对有离群值的数据更稳健，我们将使用这类检验来比较单位价格的分布。Kolmogorov-Smirnov 检验可以用来比较任何两个样本，以检查两者都来自相同分布的原假设的有效性。这个检验通过计算两者累计经验分布函数差值的最大值这个统计量来进行。如果两个分布类似，那么这个最大距离应该很小。

对于交易数量少于 20 的产品，我们将寻找与它的单位价格分布最相似的产品，然后用 Kolmogorov-Smirnov 检验来检查两个产品是否在统计意义上相似。对所有的产品组合进行这个检查，计算将会变得过于复杂。我们决定采用前面计算过的分布特征（中位数和 IQR）。也就是说，对于每一个交易数量较少的产品，我们寻找与该产品有相似中位数和 IQR 的产品。对于这种类似的产品，都进行了各自的单价分布之间的 Kolmogorov-Smirnov 检验。下面的代码用来获取一个矩阵（similar），矩阵中存储的是这种少于 20 个交易的每个产品的检验信息。它用对象 ms 来保存前面获得的每个产品的单位价格的中位数和 IQR。

```
> dms <- scale(ms)
> smalls <- which(table(Prod) < 20)
> prods <- tapply(sales$Uprice, sales$Prod, list)
> similar <- matrix(NA, length(smalls), 7, dimnames = list(names(smalls),
+     c("Simil", "ks.stat", "ks.p", "medP", "iqrP", "medS",
+         "iqrS")))
> for (i in seq(along = smalls)) {
+     d <- scale(dms, dms[smalls[i], ], FALSE)
+     d <- sqrt(drop(d^2 %*% rep(1, ncol(d))))
+     stat <- ks.test(prods[[smalls[i]]], prods[[order(d)[2]]])
+     similar[i, ] <- c(order(d)[2], stat$statistic, stat$p.value,
+         ms[smalls[i], ], ms[order(d)[2], ])
+ }
```

上面代码首先对对象 ms 的数据进行标准化，避免计算距离时负值的影响。在初始化后，主循环对每一个有少数交易的所有产品进行循环，循环的前两个命令计算所分析产品的分布特性和所有其他产品之间的距离（i 的当前值）。产生的结果对象（d）含有这些距离值。第二个最小距离对应的产品是与正在考虑的产品最类似的产品。这里是第二最小距离，因为第一最小距离是产品本身。我们再次注意到，使用产品单位价格的中位数和 IQR 信息来计算产品之间的相似性。下一步是进行 Kolmogorov-Smirnov 检验，来比较两个单位价格的分布。这里调用函数 ks. test( )。这个函数返回显著性信息，其中“提取”了检验的统计量值和各自的显著性水平。统计量取值是两个累积经验分布函数差值的最大值。接近 1 的置信度水平值表示有很强的统计证据表明两个分布相等这一原假设是正确的。下面我们给出结果矩阵 similar 对象的前几行：

```
> head(similar)

    Simil   ks.stat       ks.p     medP      iqrP     medS      iqrS
p8   2827 0.4339623 0.06470603 3.850211 0.7282168 3.868306 0.7938557
p18   213 0.2568922 0.25815859 5.187266 8.0359968 5.274884 7.8894149
p38  1044 0.3650794 0.11308315 5.490758 6.4162095 5.651818 6.3248073
p39  1540 0.2258065 0.70914769 7.986486 1.6425959 8.080694 1.7668724
p40  3971 0.3333333 0.13892028 9.674797 1.6104511 9.668854 1.6520147
p47  1387 0.3125000 0.48540576 2.504092 2.5625835 2.413498 2.6402087
```

行名称表明在为其寻找最相似的产品。第一列有它最相似产品的信息。可以用下面代码得到矩阵 similar 前面几行相应产品的 ID：

```
> levels(Prod)[similar[1, 1]]
[1] "p2829"
```

179 ~ 182

在 Kolmogorov-Smirnov 统计量和置信度水平列之后，我们有产品的中位数、IQR 和最相似产品。在 90% 显著性水平下，我们可以检查单位价格分布有相似性的产品的数量：

```
> nrow(similar[similar[, "ks.p"] >= 0.9, ])
[1] 117
```

或者，更有效率的方式，

```
> sum(similar[, "ks.p"] >= 0.9)
[1] 117
```

正如你从少于 20 个交易的 985 个产品中所看到的，我们只能为其中的 117 个产品找到类似的产品。不过，当分析哪些交易是不正常交易时，这些都是有用的信息。对于这 117 个产品，我们可以把更多的交易纳入决策过程中，从而提高检验的统计显著性水平。我们保存了 similar 对象，以后需要产品之间的这种相似性时可以直接使用该对象：

```
> save(similar, file = "similarProducts.Rdata")
```

4.3 定义数据挖掘任务

有些交易报告被强烈怀疑为欺诈交易，这个应用程序的主要目的是运用数据挖掘工具，为确定是否核查这些交易提供指导。由于可用于该检查任务的资源是有限的并且是变化的，因此这个指导应该以欺诈概率排序的形式给出。

4.3.1 问题的不同解决方法

数据集有一列（Insp）含有先前检验活动的信息。这里的主要问题是大多数可用的报告没有被检验。从确定已有报告是否为欺诈的任务角度来看，变量 Insp 中的 unkn 值和缺失值的意义是一样的。这个值代表缺少这笔交易是 OK 还是欺诈的信息。也就是说，我们的数据集有两种类型的观测值。我们有一个较小的数据集，其中的数据有标记，它有交易的特征描述和检验的结果。另外一个较大的数据集是没有标记的，它们没有被检验，列 Insp 的值为 unkn。在这种情况下，183取决于用于建模的观测值的类型，我们有不同的建模方法。

4.3.1.1 无监督技术

对于未被检验的报告，Insp 列没有任何信息，所以它对分析没有影响。对于这些观测值，我们只有对交易的描述。这意味着这些销售报告仅仅有描述它的自变量。这种类型的数据适用于非监督学习技术。这些方法被这么命名是因为它们的目标不是像有监督的方法那样在“老师”的监督下学习概念。有监督方法应用的数据就是它们所学习概念的例子（例如，欺诈或者正常交易的概念）。这就要求领域专家预先就目标概念对数据进行分类。而检验结果未知的报告则不适用于这种情况。因此我们面临的不是预测任务（即有监督方法的目标），而是一个描述性的数据挖掘任务。

聚类分析是描述性数据挖掘的一个例子。聚类方法试图对一组观测值形成多个聚类，一个聚类内的个案相似，从而找到这些观测值的“自然”组别。相似性概念通常要求由描述观测值的变量所定义的空间给出一个距离定义。这个距离定义是衡量一个观测值和其他观测值之间距离的函数。距离靠近的个案通常认为属于同一个自然组。

异常值检测也可以看做是描述性的数据挖掘任务。有些异常值（或称为离群值）检测方法假定数据的预期分布，把背离这一分布的任何值标记为异常值。另一个常见的异常值检测



策略是假定一个变量空间的距离度量，然后把距离其他观测值“太远”的观测值标记为异常观测值。

从以上的描述中我们可以知道在聚类和异常值检测之间有很密切的关系。基于观测值之间距离的方法都是这样的。根据定义，异常值是十分不同的个案，它们与聚类中其他观测值距离太远，因此不能和聚类中的其他观测值一致。这意味着一个好的数据集聚类不应该在大的数据类中含有异常值。至多我们期望该异常值和其他异常值类似，但根据定义，这些是罕见的观测值，因此不应该形成一个大的数据类。

我们问题中应用的无监督技术有一些约束条件。事实上，我们的目标是得到一组观测值的异常值排序。这个排序作为公司内检验决策的基础。这意味着选择的无监督工具必须可以用来识别和排列异常值。4.4.1节描述了我们用来解决这个数据挖掘任务的无监督技术。

无监督学习的参考文献

聚类分析是一个成熟的研究方法。好的参考文献是 Kaufman 和 Rousseeuw（1990）、Murtagh（1985）的著作。更多有关数据挖掘方面的内容可以在很多数据挖掘的参考书中找到，例如 Han 和 Kamber（2006）的书。很多学科中都有异常值探测的研究。标准的参考资料包括 Barnett 和 Lewis（1994）和 Hawkins（1980）的书。Austin（2004）和 Chandola 等人（2007）给出了关于异常值探测很好的概述。有关聚类和异常值之间的关系，可以参考 Ng 和 Han（1994）、Torgo（2007）的文章。

184

4.3.1.2 有监督技术

有正常或者欺诈标记的交易（经过检验）集合可以应用其他类型的建模方法。有监督学习方法应用这种有标记的数据类型。有监督学习方法的目标是找到目标变量（需要学习的概念）和一组自变量（预测变量或者属性）之间的关系。这个模型可以认为是一个与描述了目标变量 Y 和预测变量 X_1，X_2，…，X_p 之间关系的未知函数 $Y=f(X_1, X_2, \cdots, X_p)$ 的近似。这个建模技术的任务是得到能够优化某些选择准则的模型参数，例如，最小化模型误差。这个搜索任务在学习现象的观测值样本的帮助下进行，即它是基于包含所学习概念例子的数据集。这些例子是变量 X_1，X_2，…，X_p，Y 的特殊实例。如果目标变量 Y 是连续的，那么就有一个（多元）回归问题。如果 Y 是一个名义变量，那么就有一个分类问题。

就我们的数据集而言，目标变量是检验任务的结果，它有两个可能取值：ok 和 fraud。这意味着我们的目标是学习欺诈报告和正常报告的概念。因此我们面临一个分类问题。注意，由于我们不能确定报告是否为欺诈，所以没有经过检验的交易是不能用在这些任务中的。这意味着如果我们想要用分类技术得到一个模型，然后用它来预测一个给定的报告是不是欺诈，那么我们只能用401 146个报告中的15 732个报告作为训练样本。

我们面对的分类问题有影响性能评估和模型本身的特殊性。这个特殊性在于，在两个可能的类值中，一个值的出现频率远远大于另一个值。实际上，在15 732个检验报告中，14 462个是正常交易，剩下的只有1270个报告为欺诈。此外，这个不太频繁的概念实际上是这个问题中最重要的概念，因为它涉及应用的目的：欺诈侦测。这意味着我们必须选择这样的评价标准，它能够正确地衡量不频繁分类的性能评价，同时我们应该选择的建模技术也必须能够处理非常不平衡的数据。

本案例对分类工具的应用进行了一些改变。事实上，我们感兴趣的是按照可能存在欺诈的概率大小对交易排序。这意味着对于新报告组成的测试集，我们将用模型决定哪些报告需要进行检验。对于给定的测试个案，有些分类算法只能够输出该个案的类别标签。这对于我们的问题是不够的，因为它没有给出划分为欺诈的个案的排序。如果有太多这样的个案，而检验资源有限，我们就不能决定先处理哪一个。我们需要的是一个概率分类，也就是说，模型不仅应该预测

185

一个分类标签，而且还应该有这个标签的相关概率。这将使我们按照所估计的报告为欺诈的概率来对个案进行排序。

有监督学习的参考文献

有监督学习（或者称为预测模型）是一门成熟的学科，它有很多不同的方法来获得未知预测函数的逼近。任何关于数据挖掘的书都包括这种技术（例如，Han 和 Kamber（2006）、Hand 等（2001），或者 Hastie 等（2001））。类的不平衡问题也是许多研究工作的主题，例如，Chawla（2005）或者 Kubat 和 Matwin（1997）。

4.3.1.3 半监督技术

有时候找到有标记观测值的成本很大，也就是说有目标变量取值的个案很难找到，因此有了半监督方法。这种信息通常需要领域专家的工作，这增加了数据收集的成本。另一方面，特别是随着传感器和其他类型数据自动采集装置的广泛使用，没有标签的数据往往是容易获得的。因此，经常遇到的问题是很大比例的数据没有标记，而只有少量的带标记数据和它们一起。正如前面所看到的，这是本章中应用存在的情况。

半监督技术之所以这样命名是因为它可以处理这种存在有标记个案和未标记个案的数据集。通常有两种不同的半监督方法。一方面，一种半监督分类技术是借助未标记个案提供的额外信息来提高标准的监督分类算法的性能。另一种半监督方法是半监督聚类方法，它尝试在聚类过程中形成组别的准则上包含一些约束，这些约束是基于标记的数据，这样就可以对聚类过程进行纠偏。

半监督聚类的思想利用现有的标签来对聚类过程纠偏，使相同标签的个案在同一类中（must-link 约束），或者把不同标签的个案放入不同的组中（cannot-link）。在基于搜索的半监督聚类中，改变形成聚类的标准来对方法纠偏，从而找到合适的个案聚类。在基于相似性的半监督方法中，对算法使用的指标进行优化以满足标记数据所施加的限制。这意味着，通过“歪曲”距离的概念来反映 must-link 和 cannot-link 约束。

186

半监督分类方法有很多的替代方法。一个众所周知的方法是自我训练。这是一个迭代方法，它是从获得有标记数据的分类模型开始。下一步使用这个模型来对无标记数据进行分类。把模型具有较高分类置信度的个案和预测标记一起加入到原始训练集，从而扩大了训练集。使用这个新的训练数据集，重新获取模型，然后重复以上整个过程，直到满足一定的收敛准则。半监督分类模型的另一个例子是直推式支持向量机（TSVM）。TSVM 的目标是获得一个未标记数据集的标记，从而使线性边界在初始标记数据和未标记数据上达到最大的利润（参见 3.4.2.2 节）。

另外，我们应该考虑应用的特殊限制，即获得离群值的排序。取决于我们是使用半监督聚类方法还是半监督分类方法，可以使用与前面给出的无监督和有监督方法同样的策略相应地来完成我们的任务。

半监督方法的参考文献

半监督学习方法在研究上受到了越来越多的重视。Zhu（2006）、Seeger（2002）和 Zhu（2005）对已有工作给出了很好的概述文章。

4.3.2 评价准则

本节讨论怎样评价模型。当给出交易报告的一个测试集时，每个模型将产生这些报告的排序。本节讨论怎样评价这个排序。

我们还描述了用于获取选定评价指标的可靠估计的实验方法。

我们数据集的特性是同时包括标记和未标记数据。在这个应用中，它们被转化为检验和未检验的报告。这增加了模型比较的困难，因为对监督和无监督方法的评价方法通常是不同的。模型得到的排序很可能同时包括标记和未标记的观测值。对前者而言，很容易评价将标记数据列

入待检查报告集合的正确性。在未标记的情况下，因为我们不能确定这些个案是否为欺诈，所以这种评价是比较困难的。 187

4.3.2.1 决策精确度与回溯精确度

对我们的应用而言，一个成功模型应该得到一个交易排序，其中已知的欺诈交易在排序的顶部。我们的数据中有欺诈的报告占少数。给定一个我们的资源所允许检验的报告个数 k，我们希望在排序的顶部 k 个位置中，或者只有欺诈交易的报告或者未检验的报告。然而，我们希望这个测试集中所有已知的欺骗报告出现在这 k 个位置中。

正如我们在3.3.4节已经看到的，当我们的目标是预测一个小集合的罕见事件（这里是欺骗）时，决策精确度和回溯精确度是合适的评价指标。给定检验限制 k 值，我们可以计算排序的最顶端 k 个位置的决策精确度和回溯精确度。这个限定值 k 决定了根据模型哪些报告应该被检验。从监督分类的角度看，这相当于把顶端的 k 个位置预测为 fraud 类，其余的则为正常报告。决策精确度告诉我们顶端 k 个值的多大比例事实上是标记为欺诈的报告。回溯精确度的值给出这 k 个位置所包含的测试集的欺诈行为的比例。我们应该注意到，所获得的值是悲观的。实际上，如果顶端 k 个值包括的是未标记的报告，它们不会进行决策精确度和回溯精确度的计算。但是，如果对它们进行了检验，我们就可以发现它们实际上是诈骗交易，因此决策精确度和回溯精确度可能会更高。

通常决策精确度和回溯精确度之间有一个权衡。比如，如果所有测试集个案都预测为欺诈事件，那么很容易使决策精确度达到100%。然而，这样的策略也将不可避免地导致非常低的回溯精确度。但是，我们当前的应用有一些特殊性。给定用于检验行为的资源的约束条件，我们真正想要的是最大限度地利用这些资源。这意味着，如果我们可以用 x 小时检查报告，并能够在这 x 小时捕捉到所有的欺诈行为，那么我们很高兴！即使在这 x 小时，我们实际上也检查了一些正常的报告，也就是说有较低的回溯精确度。回溯精确度实际上是在此应用中的关键问题。我们需要的是能够用现有的资源达到100%回溯精确度。

4.3.2.2 提升图和PR曲线

4.3.2.1节提到计算给定检验行为的决策精确度和回溯精确度。在不同的检验水平下查看模型的性能是一件有趣的事情。不同的模型可能会在不同的水平上占据一定的优势，而且在比较模型时，这也可能成为有用的信息。

决策精确度/回溯精确度（PR）曲线是模型性能对这两者的一种可视化表示。通过在不同的工作点得到上面两个统计量的插值，从而得到该曲线。这些工作点由模型提供的感兴趣的类别排序的中断点给出。在我们的例子中，这将对应于应用在模型所产生的离群值排序上的不同的资源限制。对不同的限制水平（即检验更少或更多的报告）进行迭代，得到不同的决策精确度和回溯精确度。PR曲线允许这种类型的分析。 188

添加包 ROCR（Sing et al.，2009）包含多个函数，这些函数对评估二进制分类（即我们上文所提到的有两个分类的问题）非常有用。这是一个额外的添加包，必须安装后才能使用以下的程序代码。该添加包实现了很多评价指标，而且还包括如何获得很多曲线的方法。PR曲线可以很容易地通过该添加包中的函数得到。使用添加包非常简单，我们使用模型的预测功能和测试集的真实值得到类 prediction 的一个对象。这是通过函数 prediction() 实现的。由此产生的结果对象可以当做函数 performance() 的参数值，以获得一些评价指标。最后，将函数 performance() 的运算结果用于 plot() 函数，画出不同的性能指标曲线。以下代码应用该添加包的一些示例数据来说明这个过程：

```
> library(ROCR)
> data(ROCR.simple)
> pred <- prediction(ROCR.simple$predictions, ROCR.simple$labels)
> perf <- performance(pred, "prec", "rec")
> plot(perf)
```

运用上述代码画出的 PR 曲线如图 4-5 的左图所示。添加包 ROCR 产生的 PR 曲线具有锯齿形状。这通常认为是不明确的，有一种方法可以克服这种情况。也就是说，在一个确定的回溯精确度 r 下，可以计算任何大于或等于 r 的回溯精确度水平所对应的决策精确度的最大值 $\mathrm{Prec}_{\mathrm{int}}$，并把该值作为水平 r 的决策精确度，如式（4-1）所示。

$$\mathrm{Prec}_{\mathrm{int}}(r) = \max_{r' \geq r} \mathrm{Prec}(r') \tag{4-1}$$

如果我们仔细地观察 performance( ) 函数的返回值，我们就会发现有一个名为 y. values 的属性，它含有图形 y 轴上的值，即所绘制的决策精确度。我们依据式(4-1) 计算插值的决策精确度，并用它替换 y 轴坐标值，就可以得到一个无锯齿效果的图。下面的代码就实现了这种理想的图形：

```
> PRcurve <- function(preds, trues, ...) {
+     require(ROCR, quietly = T)
+     pd <- prediction(preds, trues)
+     pf <- performance(pd, "prec", "rec")
+     pf@y.values <- lapply(pf@y.values, function(x) rev(cummax(rev(x))))
+     plot(pf, ...)
+ }
```

上述代码运用了函数 lapply( )，因为实际上，属性 y. values 的值是一个列表，它包括了实验过程中多个迭代的结果。我们将在本章的后面利用这个事实。对于每一个决策精确度向量，用函数 cummax( ) 和函数 rev( ) 来计算插值的预测精确度。后一个函数仅仅是将向量倒序排列，而函数 cummax( ) 的功能是获得一组数据的累计最大值。如果你难以理解这一概念，可以用一组向量数据来实验该函数。函数 PRcurve( ) 已经包含在本书的添加包中，所以不需要输入上述代码，直接使用就可以了。可以应用 PRcurve( ) 函数演示上面给出的示例数据，产生图 4-5 的右图：

```
> PRcurve(ROCR.simple$predictions, ROCR.simple$labels)
```

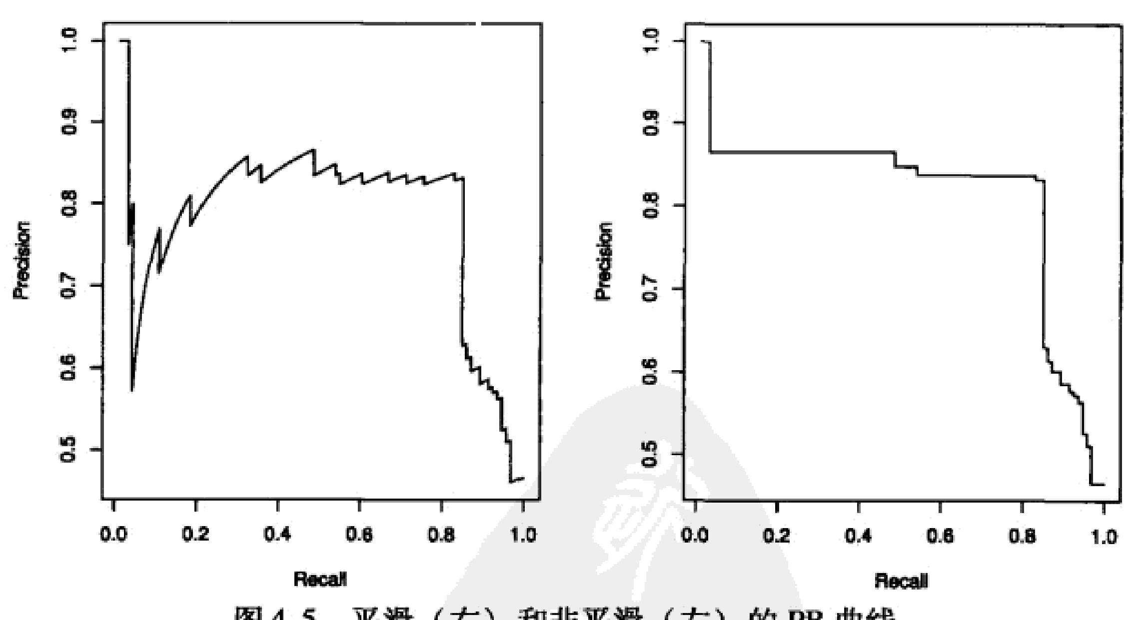


图 4-5 平滑（右）和非平滑（左）的 PR 曲线

如何用这类曲线来评价异常值排序模型？我们有一个含有变量 Insp（有 3 个可能的取值：unkn、ok 和 fraud）的测试集，以及由某些模型产生的测试集的观测值排序。我们要求模型给出测试集中每一个观测值的异常值排序分数，这些分数的范围为 0 ~ 1。分数越高，说明这个观测值是欺诈交易的模型置信度越高。这个分数是获得观测值排序的信息依据。

189 ~ 190

表 4-1　示例的混淆矩阵

		预测		合计
		ok	fraud	
真实值	ok	3	1	4
	fraud	2	1	3
合计		5	2	7

如果我们按照异常值分数对测试集记录进行排序，那么根据我们检验的资源限制 k，可以计算出不同的预测精确度和回溯精确度值。确定检验资源限制等价于选择上述我们认为一个观测值为欺诈的异常值分数的阈值。我们来看一个小例子。假设我们有 7 个测试样本，Insp 列的取值为 {ok, ok, fraud, unknown, fraud, fraud, unknown}。假设某个模型对这些观测值给出的异常值分数为 {0.2, 0.1, 0.7, 0.5, 0.4, 0.3, 0.25}。如果我们对这些分数进行排序，得到 {fraud, unknown, fraud, fraud, unknown, ok, ok}。如果检验范围只允许对两组观测值进行检验，那么将等价于模型把向量 {fraud, unknown, fraud, fraud, unknown, ok, ok} 预测为向量 {fraud, fraud, ok, ok, ok, ok, ok}。这正好对应于表 4-1 的混淆矩阵，而且以下的决策精确度和回溯精确度也是通过该混淆矩阵计算出的：

$$\text{Prec} = \frac{1}{1+1} = 0.5 \qquad \text{Rec} = \frac{1}{2+1} = 0.3333$$

值得注意的是，就像在 4.3.2.1 节中提到的，对于没有检验的报告，给出的是决策精确度和回溯精确度的悲观估计。因此，报告中排名第二的 fraud 的预测值，认为是不正确的，因为它的真实值为 unkn，我们不能肯定这个值是不是具有欺诈性。

我们将运用这种对于异常值排序的事后处理方式，去获得它们的决策精确度和回溯精确度以及相应的 PR 曲线。

提升图给我们提供了一个关于模型预测的不同角度。这些图给回溯精确度更高的重要性，因此在某种程度上，就像在 4.3.2.1 节最后提到的那样，它更适合我们的目标。这些图形的 x 轴是阳性预测率（RPP），它是模型预测一个阳性类的概率。这个值由阳性分类预测值的数量除以测试集的总观测值数量来估计。比如在表 4-1 这个例子中，这个值就是 $(1+1)/7$。在我们应用程序中，我们可以将这个统计量视为选定检验的报告比例。提升图的 y 轴是用回溯精确度除以 RPP 得到的商。

我们可以用 ROCR 包中的函数绘制提升图。以下是一个应用示例，得到提升图如图 4-6 的左图所示。

191

```
> pred <- prediction(ROCR.simple$predictions, ROCR.simple$labels)
> perf <- performance(pred, "lift", "rpp")
> plot(perf, main = "Lift Chart")
```

尽管在我们这个特定应用中并不是完全要说明提升图的有用性。一个更有趣的图将会给我们演示根据 RPP 所捕获的检验限制得到的回溯精确度值。这个曲线命名为累积回溯精确度图，这个图可以通过 ROCR 包的函数实现，其代码如下：

```
> CRchart <- function(preds, trues, ...) {
+     require(ROCR, quietly = T)
+     pd <- prediction(preds, trues)
+     pf <- performance(pd, "rec", "rpp")
+     plot(pf, ...)
+ }
```

再次应用前面的虚拟例子，得到图 4-6 的右图：

```
> CRchart(ROCR.simple$predictions, ROCR.simple$labels,
+         main='Cumulative Recall Chart')
```

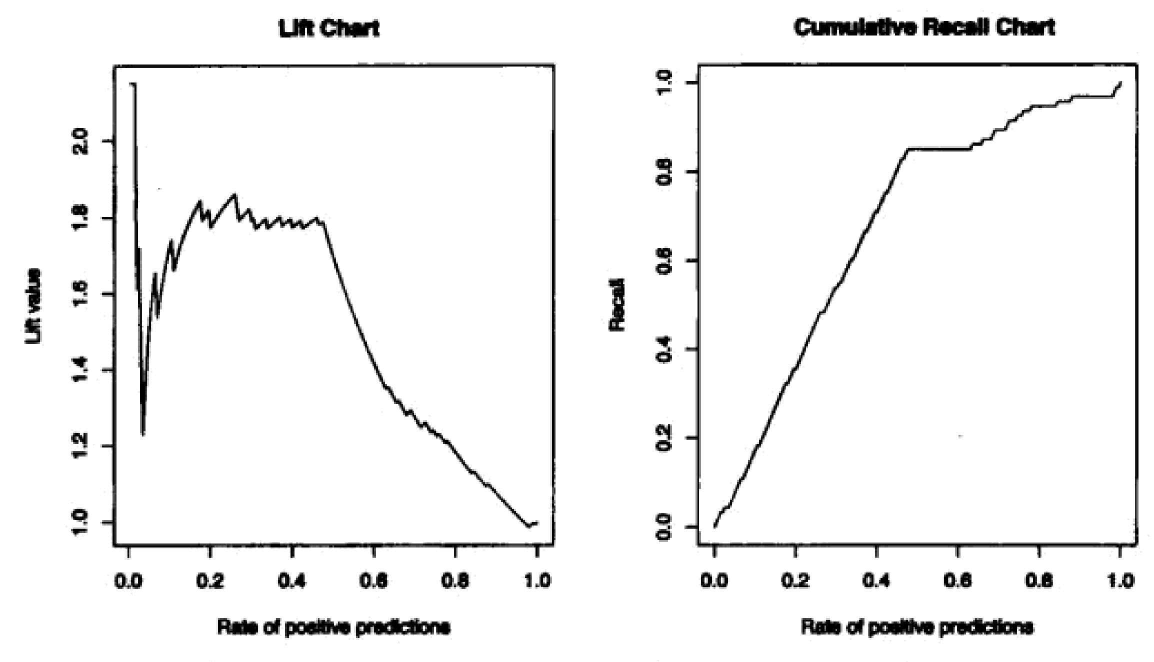


图4-6　提升图（左图）以及累积回溯精确度图（右图）

对累积回溯精确度图而言，模型的曲线越靠近左上角，模型越好。本书的添加包中有函数 192 CRchart()，只要安装了该包，任何时候都可以使用它。

4.3.2.3　标准价格的标准化距离

在前面的章节中我们看到的性能评价方法，仅是对有标记报告的排序进行评价。它们是监督分类的评价指标。由模型得到的这些排名的前面位置极有可能包含未标记的报告。这些未标记的报告是否在排序中有正确的位置呢？我们不能肯定这一点，因为我们没有对它们进行检验。不过，我们可以对它们进行浅显的分析。例如，我们可以把它们的单位价格和同一产品报告的标准价格进行比较。我们预期这些价格之间差异是较大的，这就是一个提示，说明报告中有错误。这种情况下，报告的单位价格和相应产品的标准单位价格的距离，就是模型所获得的异常值排名质量的一个较好指标。

不同的产品有不同规格的单位价格，如图4-4所示。为了避免这些不同价格对异常值排名的影响，我们对单位价格和标准价格之间的距离进行标准化。运用IQR（四分位距）来标准化这个距离：

$$NDTP_p(u) = \frac{|u - \widetilde{U}_p|}{IQR_p} \tag{4-2}$$

其中 $\widetilde{U}_p$ 是产品 p 的标准单位价格，是该产品交易的单位价格的中位数；而 IQR_p 是该产品单位价格的四分位距。

在我们的实验中，我们使用 $NDTP_p$ 的平均值作为评价模型性能的一个指标。以下的程序代码用来计算这个统计量：

```
> avgNDTP <- function(toInsp,train,stats) {
+   if (missing(train) && missing(stats))
+     stop('Provide either the training data or the product stats')
+   if (missing(stats)) {
+     notF <- which(train$Insp != 'fraud')
+     stats <- tapply(train$Uprice[notF],
+                     list(Prod=train$Prod[notF]),
```

```
+                    function(x) {
+                      bp <- boxplot.stats(x)$stats
+                      c(median=bp[3],iqr=bp[4]-bp[2])
+                    })
+     stats <- matrix(unlist(stats),
+                    length(stats),2,byrow=T,
+                    dimnames=list(names(stats),c('median','iqr')))
+     stats[which(stats[,'iqr']==0),'iqr'] <-
+         stats[which(stats[,'iqr']==0),'median']
+   }
+
+   mdtp <- mean(abs(toInsp$Uprice-stats[toInsp$Prod,'median']) /
+                stats[toInsp$Prod,'iqr'])
+   return(mdtp)
+ }
```

上述函数是把模型所选择的用于检验的交易作为主要参数。其次，它必须接收训练集数据以获得每个产品的中位数和四分位距，或者接收已经准备好了具有这一组信息的数据结构，这样可以提高反复调用此函数时的计算效率。如果接收的是训练数据，那么函数将用训练集数据计算每一个产品的无欺诈交易的中位数和四分位距。那么可能会发生四分位距为 0 的情况，特别是在产品交易量较少的情况下。为了避免计算 $NDTP_p$ 时除数为零的情况，我们把该情况下的四分位距设置为中位数的值。

4.3.3 实验方法

我们使用的数据集的大小很合理。这种情况下，我们选择保留（Hold Out）方法来进行实验比较是有意义的。这个方法把已有的数据集随机地分成两部分（通常的比例为 70% 和 30%），其中的一部分用于获取模型，而另一部分用来测试。如果有必要，这个过程可以重复多次，以确保获得的结果更稳健。我们数据集的容量可以确保我们获得的结果在统计学上是可靠的。如果我们选择 30% 的数据作为验证集，就相应于 120 343 个报告。

在这种情况下，困难就是不同类型报告之间分布的不均衡性，即在标记个案上是不平衡的。正常的重抽样策略可能会导致一个测试集的正常报告/欺诈报告的不同分布。当有这种不均衡的分布类型时，推荐使用分层抽样方法。这种方法从具有不同类型的观测值袋子中随机抽样，以确保所抽取的样本遵守初始数据的分布。例如，如果有 10% 的个案属于类 X，其余的 90% 都是属于类 Y，就把所有的观测值放入两个独立的袋子中。如果想要一个有 100 个样本的随机分层抽样，就从有类 X 的观测值袋子中随机抽取 10 个样本，然后剩余的 90 个样本从有类 Y 观测值的袋子中抽取，这样做就遵守了最开始的类比例。

在本书的 R 添加包中，有一个 holdOut( ) 函数，它的作用和第 3 章中的交叉验证和蒙特卡罗实验类似，用于进行 hold-out 实验。该函数的一个参数，是 hldSetting 类对象，它用于设置实验。在其他参数中，这个对象允许你指定应用分层抽样方法。4.4 节给出了几个应用该函数来获取我们选定的评价统计量的 hold-out 估计值的例子。这些选定的统计量是决策精确度、回溯精确度和平均 NDTP。下面的函数用来计算这些指标：

193 ≀ 194

```
> evalOutlierRanking <- function(testSet,rankOrder,Threshold,statsProds) {
+   ordTS <- testSet[rankOrder,]
+   N <- nrow(testSet)
+   nF <- if (Threshold < 1) as.integer(Threshold*N) else Threshold
+   cm <- table(c(rep('fraud',nF),rep('ok',N-nF)),ordTS$Insp)
+   prec <- cm['fraud','fraud']/sum(cm['fraud',])
+   rec <- cm['fraud','fraud']/sum(cm[,'fraud'])
```

```
+   AVGndtp <- avgNDTP(ordTS[nF,],stats=statsProds)
+   return(c(Precision=prec,Recall=rec,avgNDTP=AVGndtp))
+ }
```

该函数需要用户提供测试集、模型产生的该测试集的排序、一个指定检测限值的阈值（无论是百分比或报告的数量）和产品的统计量（中位数和四分位距）。

在4.2.3.2节中，我们观察到的产品相当不同，实际上有些产品有很少的交易。因此，我们可以质疑一起分析所有产品的交易是否有意义。支持一起检查它们是因为有一个变量（产品ID）可以用来区分产品，因此如果有必要，建模技术可以使用该变量。此外，把所有交易放在一起，模型可以利用这些产品之间的关系。然而，另一种是依次分析每一个产品，获得它们相应交易的离群值评分排序。这就需要一个额外的步骤，即从单个产品的排名获得最终的全局排名，但该步骤比较简单。我们将会对应用不同策略建立的模型进行实验。从实验方法的角度来看，我们把所有的产品聚在一起。在这些交易中，我们将用分层保留策略来随机选择一个测试集。这个测试集将应用于不同的建模技术，它们将返回按照这些交易为欺诈的概率估计的排序。本质上，模型可以决定对产品进行单个分析或一起分析所有产品。

4.4 计算离群值的排序

本节描述用于获取离群值排序的不同模型。对于每一次尝试，我们都采用70%/30%的分层保留策略来估计结果。

195

4.4.1 无监督方法

4.4.1.1 修正的箱图规则

在4.2.2节中，我们定义了箱图规则，用于侦测一个服从近正态分布的连续变量的离群值。例如，产品的单位价格这个例子。基于此，我们可以把这个简单的规则看做是可以应用于本章数据的基准方法。

应用箱图规则发现每一个产品交易中的异常单位价格，从而识别出一些可能是离群值的数值。我们可以把一个规律用于任何一个出现在已给测试集中的产品交易集合上。最后，我们得到每一个产品可能的离群值集合。我们需要决定怎么从这些离群值组合得到整个测试集的离群值排序。这意味着为了排序，我们必须能够清晰地区分出离群值。一个可能的方法就是4.3.2.3节描述的标准化到标准（中位数）单位价格（NDTP）的距离。这个方法可视为箱图规则的一个变形，因为两者都用到中心值的距离来决定一个值的“离群程度”。这种方法（NDTP）的优点在于它是一种无量纲的度量，因此可以将不同产品的数值混放在一起，从而给出一种适用于全部测试集数据的全局排序。

```
> BPrule <- function(train,test) {
+   notF <- which(train$Insp != 'fraud')
+   ms <- tapply(train$Uprice[notF],list(Prod=train$Prod[notF]),
+                function(x) {
+                  bp <- boxplot.stats(x)$stats
+                  c(median=bp[3],iqr=bp[4]-bp[2])
+                })
+   ms <- matrix(unlist(ms),length(ms),2,byrow=T,
+                dimnames=list(names(ms),c('median','iqr')))
+   ms[which(ms[,'iqr']==0),'iqr'] <- ms[which(ms[,'iqr']==0),'median']
+   ORscore <- abs(test$Uprice-ms[test$Prod,'median']) /
+              ms[test$Prod,'iqr']
+   return(list(rankOrder=order(ORscore,decreasing=T),
+               rankScore=ORscore))
+ }
```

上述函数的参数是训练集和测试集。在计算出每一种产品的中位数和四分位距（IQR）后，这个函数就用这些统计量利用式(4-2）来计算离群值分数。最后，它返回一个由离群值分数和测试集观察值排序所组成的列表。假设这个方法用 NDTP 值来得到排序，那么可预见这个指标的平均值可以很好地进行评分。

注意，这里是我们应该应用产品之间相似性信息的地方。实际上，对于交易量不多的产品，我们可以考虑检验是否有产品的单位价格分布显著地与该产品相似。如果存在这样的产品，就把它的交易加入，因此可以用一个较大的样本来估计中位数和 IQR 统计量。这可以通过调用函数 tapply( ) 来实现，这里我们包括有关相似产品的信息，它们存储在文件“similarProducts. Rdata”中（参见 4. 2. 3. 2 节的最后）。这一点留给读者自行练习。

196

我们现在用保留实验方法来评价这个简单的方法。为了计算每种产品的平均 NDTP 值，首先计算每一种产品的中位数和 IQR 值。然后我们用所有可用的数据来进行以上计算，因为我们旨在得到这些值最精确的估计，从而能够正确评价模型的离群值排序能力。因为全局信息没有传递给建模技术，所以这里不能认为是从测试数据给出信息（避免模型和测试集相关）。这只是得到模型侦测异常值能力更可靠的估计方式。

```
> notF <- which(sales$Insp != 'fraud')
> globalStats <- tapply(sales$Uprice[notF],
+                       list(Prod=sales$Prod[notF]),
+                       function(x) {
+                         bp <- boxplot.stats(x)$stats
+                         c(median=bp[3],iqr=bp[4]-bp[2])
+                       })
> globalStats <- matrix(unlist(globalStats),
+                  length(globalStats),2,byrow=T,
+                  dimnames=list(names(globalStats),c('median','iqr')))
> globalStats[which(globalStats[,'iqr']==0),'iqr'] <-
+    globalStats[which(globalStats[,'iqr']==0),'median']
```

holdOut( ) 函数需要调用一个过程，从而给实验过程的每一次迭代获得和评估 BPrule 方法。在前面几章中的交叉验证和蒙特卡罗实验中，我们为其他学习系统建立了类似的用户函数。这些函数返回一个向量，其值为给定训练集和测试集的模型评估统计量的值。这次我们需要其返回更多的信息。为了绘制 PR 图，以及累积回溯精确度曲线，ROCR 添加包函数还需要知道每一个测试样本观测值的预测值和真实值。因此也需要函数返回这些数值（预测值和真实值），以便后面绘制曲线。4. 3. 2. 2 节用一个虚拟的小例子演示了绘制图形所要应用的信息。下面代码给出的函数将在 holdOut( ) 中调用，它将返回评价统计量的值以及带有预测值和真实值信息的属性。

```
> ho.BPrule <- function(form, train, test, ...) {
+   res <- BPrule(train,test)
+   structure(evalOutlierRanking(test,res$rankOrder,...),
+             itInfo=list(preds=res$rankScore,
+                         trues=ifelse(test$Insp=='fraud',1,0)
+                        )
+            )
+ }
```

大多数 R 对象可以有附带的属性。实际上，它是把一个其他对象附加到前一个对象中，这些附加对象给前一个对象额外信息（例如维数等）。在这个例子中，我们给向量附加了 BPrule 方法的分数，它含有预测分数和预测分数相应的真实值。函数 structure( ) 用来建立一个对象并指定该对象的属性值。这些属性要有一个名字并包含一个 R 对象。这里我们用 structure( ) 函数创建一个带有 itsInfo 属性的对象。实验过程中的每一次迭代，holdOut( ) 函数保存这些属性信息。

为了存储这些信息，需要调用带有可选择参数 itsInfo = T 的 holdOut( ) 函数。这样就能确保那些所谓的“用户定义”的函数无论返回什么名为 itsInfo 的属性结果，它们都会被收集在一个列表中，并且返回为一个带有 itsInfo 属性的结果作为函数 holdOut( ) 的结果。

我们已经准备好运行 holdOut( ) 函数来得到 BPrule 系统选定的统计量的值。我们将会使用 70%/30% 分层抽样策略来对整个数据集进行划分，并对一个预先定义的检验限制值 10% 来计算决策精确度和回溯精确度。在某种意义下，预定义的 10% 检验限制条件的设定是随意的，也可以选择其他的限制值。该系统对不同限制值的全局性能由 PR 和累积回溯精确度曲线给出。我们将对以上实验过程重复三次，基于此得到保留估计。

```
> bp.res <- holdOut(learner('ho.BPrule',
+                           pars=list(Threshold=0.1,
+                                     statsProds=globalStats)),
+                   dataset(Insp ~ .,sales),
+                   hldSettings(3,0.3,1234,T),
+                   itsInfo=TRUE
+                   )
```

设置函数 hldSettings( ) 的第四个参数为 TRUE，表明使用分层随机抽样。其他参数规定了重复次数、保留集（hold-out）个案所占的百分比、随机数种子等。

本次实验结果的总结如下：

```
> summary(bp.res)

= Summary of a Hold Out Experiment ==

Stratified  3 x 70 %/ 30 % Holdout run with seed =  1234

*Dataset ::  sales
*Learner  ::  ho.BPrule  with parameters:
        Threshold  =  0.1
        statsProds  =  11.34  ...

*Summary of Experiment Results:

          Precision    Recall     avgNDTP
avg     0.016630574 0.52293272 1.87123901
std     0.000898367 0.01909992 0.05379945
min     0.015992004 0.51181102 1.80971393
max     0.017657838 0.54498715 1.90944329
invalid 0.000000000 0.00000000 0.00000000
```

结果的决策精确度和回溯精确度都是相当低的。平均只有 52% 的已知欺诈包含在用 BPrule 给出的报告欺诈排序的前 10% 的位置中。低的回溯精确度意味着 10% 的检验努力可能不能包含所有的欺诈交易，但是考虑到测试集中欺诈的比例和很低的决策精确度值，这是不正确的。极低的回溯精确度值意味着该方法在排序的顶部大部分放入的是 unkn 或者 ok 报告。考虑到 NDTP 的相对高分值，我们至少可以确定这些顶部报告的单位价格和标准价格是不相同的。事实上，NDTP 的平均值为 1.8，意味着同一种产品报告中的单位价格和中位数价格大约为这些价格 IQR（四分位距）的 1.8 倍。假定 IQR 包括 50% 的报告，它意味着这些交易是很异常的。

为了获得 PR 图和累积回溯精确度图，我们需要有方法在每一次保留（hold-out）重复上的离群值分数，以及真实的“类”标签。我们所调用的函数通过应用每一次迭代（ho. BPrule( )）的排序方法返回这些值作为统计量向量的属性。函数 holdOut( ) 把每一次迭代得到的这些额外信息收集在一个列表中。该列表作为函数 holdOut( ) 产生对象的 itsInfo 属性返回。为了获得绘图函

数所要求的格式信息，需要一些额外的步骤。以下代码的结果即为显示在图 4-7 中的曲线。

```
> par(mfrow=c(1,2))
> info <- attr(bp.res,'itsInfo')
> PTs.bp <- aperm(array(unlist(info),dim=c(length(info[[1]]),2,3)),
+                  c(1,3,2)
+                 )
> PRcurve(PTs.bp[,,1],PTs.bp[,,2],
+          main='PR curve',avg='vertical')
> CRchart(PTs.bp[,,1],PTs.bp[,,2],
+          main='Cumulative Recall curve',avg='vertical')
```

第一条语句把图窗口分成两个，这样可以并排显示两个可视化图形。第二条语句用函数 attr( ) 提取每一次迭代中 ho. BPrule( ) 返回的包含预测值和真实值信息的列表。这个函数可以用对象的任何属性名来获取该属性的信息。然后该列表被转换成一个 3 维数组。第一维是测试个案，第二维是保留（hold-out）过程的重复次数，第三维代表数值的类型（1 是预测值，2 是真实值）。例如，值 PTs. bp[3,2,1]代表这是第三个个案在保留（hold-out）过程的第二次迭代的模型预测值。函数 aperm( ) 可以用来变换数组的维数。如果理解这个复合语句有困难，可以试着单独依次运行每一个函数，查看它的输出结果（用取结果子集的方法来避免输出量过大，因为有的输出对象非常大）。

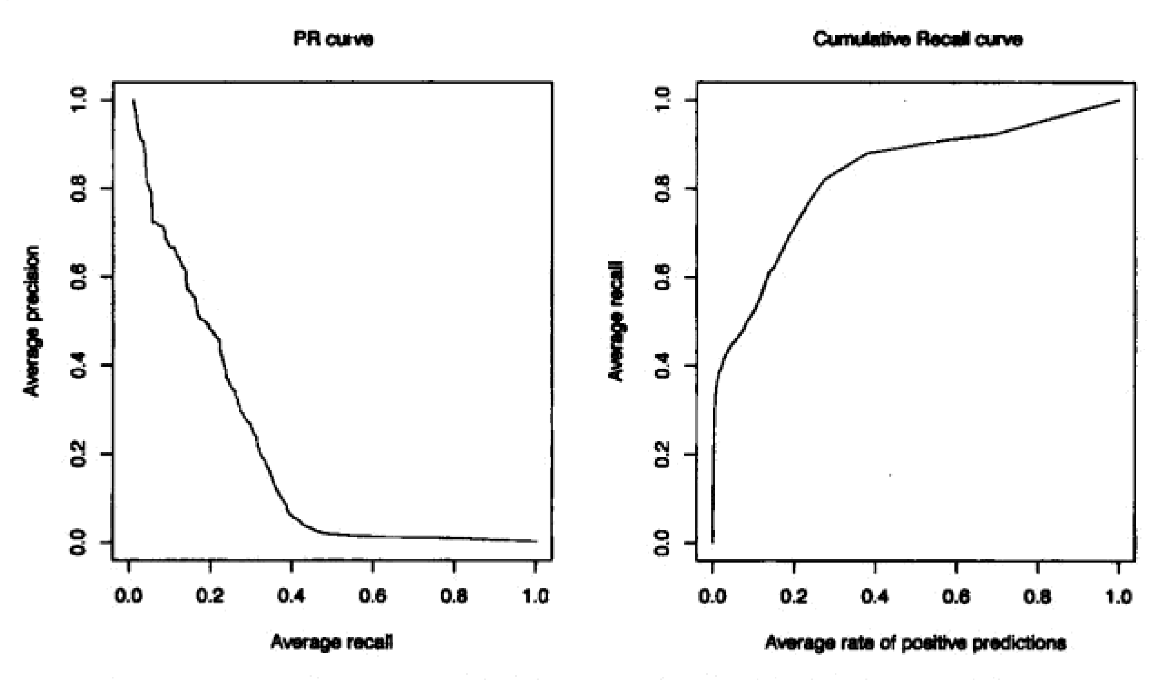


图 4-7 BPrule 方法的 PR 图（左图）和累积回溯精确度曲线（右图）

两幅曲线图都是对每次保留（hold-out）过程得到的曲线的垂直平均。累积回溯精确度曲线给出了这个方法的全局性能的视图。我们可以看到在很低的检验资源下这个方法有大约 40% 的回溯精确度。但是，为了达到 80% 的回溯精确度，大约需要检验 25% ~30% 的报告。

197 ≀ 200

4.4.1.2 局部离群值因子

离群值分类是一个有充分研究的主题。Breunig 等（2000）提出了局部离群值因子（Local Outlier Factor，LOF）系统，该方法被认为是最先进的离群值排序法。这个系统的主要思想是试着通过估计个案和它的局部邻域的分离程度来得到该个案的离群值分数。这个方法基于观察值的局部密度。在低密度区域的个案被视为离群值。个案的密度估计值是通过个案之间的距离来获得的。作者定义了一些概念并导出了计算每个点的离群分数的算法。它们是 1）一个点 p 的中心距离概念，定义为该点到第 k 个最近邻点的距离；2）个案 p_1 和个案 p_2 之间的可到达距离，该距离是 p_1 的中心距离和两个个案之间距离的最大值；3）一个点的局部可到达距离，该距离反比于 k 个可到达距离的平均值。一个个案的 LOF 是它的局部可到达距离的函数。

本书的添加包中包括了基于（Auna et al.，2009）给出的 LOF 算法的一个实现。也就是说，我们提供了函数 lofactor()，它的参数是一个数据集和计算个案 LOF 时用于指定近邻个数的 k 值。LOF 的实现仅仅限于数据集为数值型变量。实际上这也是很多模型算法的常见限制。就像我们看到的那样，我们的数据集包括了多个名义变量，这意味着我们不能直接在这个数据集上应用该函数。有多种方法可以解决这个困难。第一个方法改变 LOF 函数的源代码，使它应用混合模式的距离。有多个距离函数可以计算由不同类型变量所描述的个案间的距离。其中一个例子是添加包 cluster 中的 daisy() 函数。另一个方法对名义变量重新编码，使描述观测值的变量全部为连续型变量。任何有 n 个可能取值的名义变量可以重新编码为 $n-1$ 个二元（0/1）变量。在我们的应用中这个方法是有问题的。比如变量 ID 有 6016 个可能值，变量 Prod 有 4546 个可能值，这就意味着如果我们采用了这个策略，那么数据集将有 10 566 个变量。相比于原始数据，这是非常荒唐的增加量。所以这个方法不能够解决我们这里的问题。第三种方法单独处理每个产品，就像用 BPrule 方法那样。这样一来，不仅减轻了处理这个问题的计算机要求，也能避免应用 Prod 变量。而且，考虑到观测值之间的不同（见 4.2.3.2 节），分开处理产品总是一种看似可行的方式。因此，我们还要确定怎么处理销售人员信息（即变量 ID）。删除该变量意味着，我们认为一些异常报告是独立于它的销售人员的。这个假设看起来并不很具风险。事实上就算某些销售人员更倾向于有欺诈行为，它依然会反映在他报出的单位价格上。这种情况下，采用删除这两列（变量 ID 和变量 Prod）并分别对待每一种产品的方法比对变量重新编码的方法看起来更合理些。总之，我们对只用单位价格描述的报告数据集应用 LOF 算法：

```
> ho.LOF <- function(form, train, test, k, ...) {
+   ntr <- nrow(train)
+   all <- rbind(train,test)
+   N <- nrow(all)
+   ups <- split(all$Uprice,all$Prod)
+   r <- list(length=ups)
+   for(u in seq(along=ups))
+     r[[u]] <- if (NROW(ups[[u]]) > 3)
+                    lofactor(ups[[u]],min(k,NROW(ups[[u]]) %/% 2))
+                else if (NROW(ups[[u]])) rep(0,NROW(ups[[u]]))
+                else NULL
+   all$lof <- vector(length=N)
+   split(all$lof,all$Prod) <- r
+   all$lof[which(!(is.infinite(all$lof) | is.nan(all$lof)))] <-
+     SoftMax(all$lof[which(!(is.infinite(all$lof) | is.nan(all$lof)))])
+   structure(evalOutlierRanking(test,order(all[(ntr+1):N,'lof'],
+                                            decreasing=T),...),
+             itInfo=list(preds=all[(ntr+1):N,'lof'],
+                         trues=ifelse(test$Insp=='fraud',1,0))
+            )
+ }
```

上述函数得到了在训练集和测试集上应用 LOF 方法计算出的评估统计量。我们的方法是通过合并训练集和测试集，用 LOF 方法来对合并后的所有报告进行排序。从得到的排序中，我们选择属于测试集个案的排序分数。我们可以只对测试集进行排序，但是这样就没有用上训练数据的信息。另一个只对训练集数据排序的方法也是没有意义的，因为这是无监督的方法，其结果不能用来对测试集进行预测。

函数 split() 把全部数据按照产品来对单位价格进行分割。分割的结果是一个列表，列表的元素为相应产品的单位价格。其中的 for 循环对这些单位价格集合进行循环，应用 LOF 算法来得到每个价格的一个离群值因子。这些因子被收集在一个按照产品排列的列表（r）中。只有在至

少有三个报告时，我们才应用LOF方法，否则所有数值被标记为正常（分数为0）。在主循环之后，再次用split()函数把得到的离群值因子“添加到”数据框all中的相应交易中。下一条语句的目的在于将离群值因子改变为0~1。本书添加包中的函数SoftMax()即为实现此目的。这个函数把一群值“挤压”到0~1内。对于某些交易，函数lofactor()会出现一些Inf和NaN（无穷或无意义）值，基于这个事实，我们要约束SoftMax()函数的使用。最后，得到排序的评估分数，加上预测值和真实值一起，作为该函数的结果返回。

下一步就是就像之前对BPrule方法那样，用一个保留（hold-out）过程来获得评价指标的估计值。我们已经用了和之前一样的设置，特别是用了同一个随机数生成器种子来确保应用相同的数据分割。我们设置lofactor()函数中参数*k*的值为7。可以执行进一步的实验来调整这个参数。当执行以下指令之前，一条警告是：取决于计算机硬件，尽管是以分钟计算，但完成下面的代码可能要用很长的时间。

```
> lof.res <- holdOut(learner('ho.LOF',
+                            pars=list(k=7,Threshold=0.1,
+                                      statsProds=globalStats)),
+                    dataset(Insp ~ .,sales),
+                    hldSettings(3,0.3,1234,T),
+                    itsInfo=TRUE
+                    )
```

LOF方法的结果如下：

```
> summary(lof.res)

== Summary of a Hold Out Experiment ==

 Stratified  3 x 70 %/ 30 % Holdout run with seed =  1234

* Dataset ::  sales
* Learner  ::  ho.LOF  with parameters:
         k  =  7
         Threshold  =  0.1
         statsProds  =  11.34  ...

* Summary of Experiment Results:

          Precision     Recall    avgNDTP
avg     0.022127825 0.69595344 2.4631856
std     0.000913681 0.02019331 0.9750265
min     0.021405964 0.67454068 1.4420851
max     0.023155089 0.71465296 3.3844572
invalid 0.000000000 0.00000000 0.0000000
```

你可以观察到，对于10%的检验限制，决策精确度和回溯精确度均高于BPrule方法得到的结果。特别指出的是，回溯精确度的值已从52%增加到69%。此外，这是伴随着NDTP平均值的增加（从1.8增加到2.4）。

可以通过PR和累积回溯精确度曲线得到一个更加全局的视角。为了更好地与BPrule方法比较，我们也绘制这种方法的上述两条曲线，使用参数add = T，使得多条曲线出现在同一个图形上，如图4-8所示。

```
> par(mfrow=c(1,2))
> info <- attr(lof.res,'itsInfo')
> PTs.lof <- aperm(array(unlist(info),dim=c(length(info[[1]]),2,3)),
```

```
+                    c(1,3,2)
+                   )
> PRcurve(PTs.bp[,,1],PTs.bp[,,2],
+         main='PR curve',lty=1,xlim=c(0,1),ylim=c(0,1),
+         avg='vertical')
> PRcurve(PTs.lof[,,1],PTs.lof[,,2],
+         add=T,lty=2,
+         avg='vertical')
> legend('topright',c('BPrule','LOF'),lty=c(1,2))
> CRchart(PTs.bp[,,1],PTs.bp[,,2],
+         main='Cumulative Recall curve',lty=1,xlim=c(0,1),ylim=c(0,1),
+         avg='vertical')
> CRchart(PTs.lof[,,1],PTs.lof[,,2],
+         add=T,lty=2,
+         avg='vertical')
> legend('bottomright',c('BPrule','LOF'),lty=c(1,2))
```

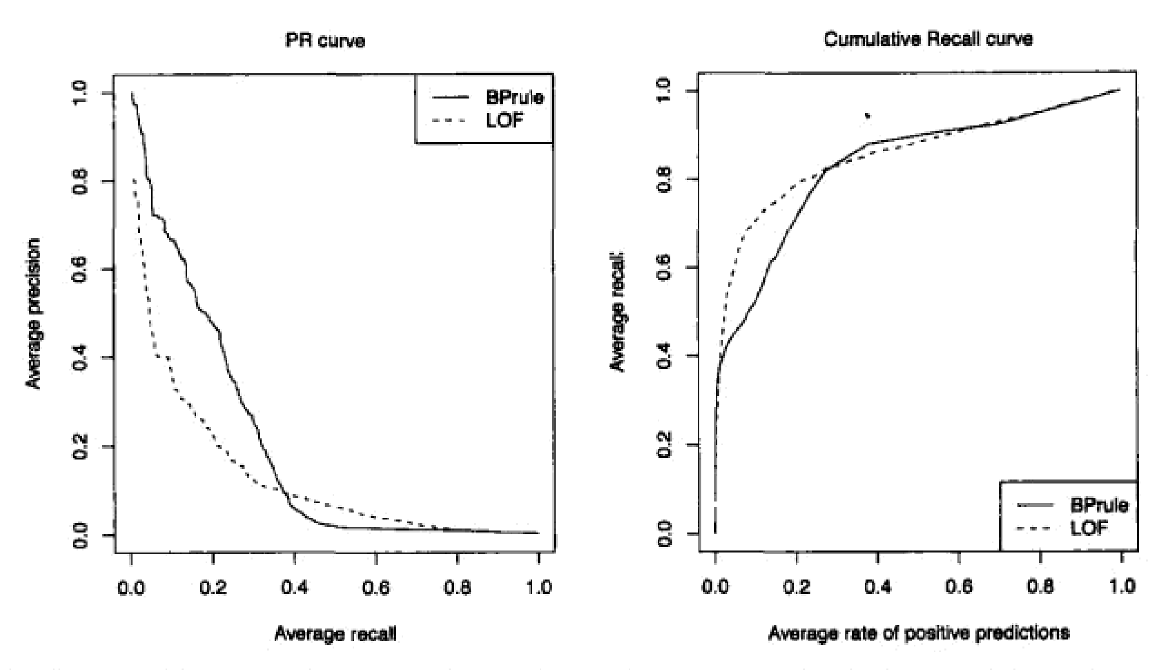


图 4-8　LOF 方法和 BPrule 方法的 PR 图（左图）和累积回溯精确度曲线（右图）

对 PR 曲线的分析（图 4-8 左图）说明，对于较小的回溯精确度值，BPrule 方法通常达到相当高的决策精确度。对于高于 40% 的回溯精确度值，虽然没有那么明显的差异，但趋势则相反。就检验限值所达到的决策精确度而言（图 4-8 的右图），可以说，对低于 25% ~30% 的检验限值，通常 LOF 方法要优于 BPrule 方法。对于高的检验限值，两者的差别就不是那么明显了，它们的结果相差不太大。鉴于该公司的兴趣在于以较低的检验努力（较低的检验限值）来降低成本（假设获得了一个好的回溯精确度），我们会对 LOF 方法更有兴趣。事实上，对于大约 15% ~20% 的检验限值，大约能侦测 70% ~80% 的诈骗行为。此外，我们应该注意到对于 10% 的检验限值，LOF 方法的 NDTP 值明显高于 BPrule 方法。

203 ~ 204

4.4.1.3　*基于聚类的离群值排名*

我们考虑的下一个离群值排名方法是基于聚类算法的结果。*基于聚类的离群值排名*（OR_h）方法（Torgo，2007）采用分层聚类法获得一个给定数据集的聚类树。聚类树是这些聚类算法合并过程的可视化表示。对这些树在不同高度水平进行切割时将给出数据的不同聚类。在最低水平，得到的类数将与训练集中观测值的个数相同。它是这些聚类迭代方法的初始解。然后该算法把前面步骤中得到的某两个类合并为一个类。这一合并过程遵循把更相似的类合并在一起的原则。当最后的两个类合并为一个由所有观测值组成的类时，迭代过程停止。聚类树描述了整个合

并的过程。基础包 stats 中的函数 hclust( ) 实现了这种类型聚类的几个变体。这个函数返回的对象包括一个数据结构（merge），该结构包括每个合并步骤涉及的那些个案的信息。OR_h 方法以这个数据结构的信息为基础进行下面的离群值排序。主要的思想法是，离群值应该不易于合并，因此当它们最终被合并时，它们合并前所属类的大小和它们被合并进去的类的大小应该相差很大。这也反映了离群值和其他观测值是很不相同的。因此把它们包含在含有更多“正常”观测值的类中将明显地降低结果组的相同性。少数时候，离群值与其他观测值的合并发生在初始阶段，但这只限于类似的离群值。否则，它们只会在聚类过程的后期合并。通常的情况是与一个更大的类合并。这是 OR_h 方法所采用的总体思路。这种方法用下面方法来计算每一个个案的离群值分数。对于每一个合并两个组（$g_{x,i}$和 $g_{y,i}$）的第 i 步，计算以下值：

205

$$\text{of}_i(x) = \max\left(0, \frac{|g_{y,i}| - |g_{x,i}|}{|g_{y,i}| + |g_{x,i}|}\right) \tag{4-3}$$

其中，$g_{x,i}$是 x 所属的组，而 $|g_{x,i}|$ 是该组的大小。请注意，因为我们感兴趣的是较小的组，所以参与合并的两个组中较大组的成员离群值分数将被赋为 0。在分层聚类算法的整个迭代过程中，每个观察值可以参与多个合并过程，有时是较大组的成员，有时是较小组的成员。数据集的每个个案的最终离群值分数由下式给出：

$$OF_H(x) = \max_i \text{of}_i(x) \tag{4-4}$$

本书添加包函数 outliers. ranking( ) 用来实现上面的算法。下面的函数使用 OR_h 方法来获取测试集中个案的离群值分数和通常的性能评价统计量：

```
> ho.ORh <- function(form, train, test, ...) {
+   ntr <- nrow(train)
+   all <- rbind(train,test)
+   N <- nrow(all)
+   ups <- split(all$Uprice,all$Prod)
+   r <- list(length=ups)
+   for(u in seq(along=ups))
+     r[[u]] <- if (NROW(ups[[u]]) > 3)
+                  outliers.ranking(ups[[u]])$prob.outliers
+               else if (NROW(ups[[u]])) rep(0,NROW(ups[[u]]))
+               else NULL
+   all$orh <- vector(length=N)
+   split(all$orh,all$Prod) <- r
+   all$orh[which(!(is.infinite(all$orh) | is.nan(all$orh)))] <-
+     SoftMax(all$orh[which(!(is.infinite(all$orh) | is.nan(all$orh)))])
+   structure(evalOutlierRanking(test,order(all[(ntr+1):N,'orh'],
+                                          decreasing=T),...),
+             itInfo=list(preds=all[(ntr+1):N,'orh'],
+                         trues=ifelse(test$Insp=='fraud',1,0))
+            )
+ }
```

206

该函数与之前的 LOF 方法很类似。我们再次使用这个方法来对产品逐个进行处理，主要是出于与之前在 LOF 方法中所描述的原因。然而，函数 outliers. ranking( ) 的参数可以接收要排序的观测值的距离矩阵，而不是数据集。这就是说，我们可以使用任何可以处理混合模式数据的距离函数来得到距离矩阵（例如，添加包 cluster 的 daisy( ) 函数）。但是，如果你决定尝试这一点，由于聚类这样大的数据集将要求大量的内存和快速的处理器，所以你将需要大量的计算资源，如集群等。即使使用这种方法对每一种产品单独处理，但在任何正常的计算机上运行下面全部保留（hold-out）实验的代码肯定需要一段较长时间。

与 LOF 方法类似，我们没有深入探索 OR_h 方法的几个参数值，只是简单地接受其默认设置：

```
> orh.res <- holdOut(learner('ho.ORh',
+                            pars=list(Threshold=0.1,
+                                      statsProds=globalStats)),
+                    dataset(Insp ~ .,sales),
+                    hldSettings(3,0.3,1234,T),
+                    itsInfo=TRUE
+                    )
```

对 OR_h 方法的结果总结如下：

```
> summary(orh.res)

== Summary of a Hold Out Experiment ==

 Stratified  3 x 70 %/ 30 % Holdout run with seed =  1234

* Dataset ::  sales
* Learner  ::  ho.ORh  with parameters:
         Threshold  =  0.1
         statsProds  =  11.34  ...

* Summary of Experiment Results:

           Precision      Recall    avgNDTP
avg     0.0220445333 0.69345072 0.5444893
std     0.0005545834 0.01187721 0.3712311
min     0.0215725471 0.67979003 0.2893128
max     0.0226553390 0.70133333 0.9703665
invalid 0.0000000000 0.00000000 0.0000000
```

207 OR_h 系统的决策精确度和回溯精确度与 BPrule 方法和 LOF 方法非常相似。对于平均 NDTP 值而言，其结果大大低于其他两个方法。

该方法的 PR 和累积回溯精确度曲线，以及以前得到的其他无监督方法的曲线，如图 4-9 所示。下面的代码是用来生成这些图形。

```
> par(mfrow=c(1,2))
> info <- attr(orh.res,'itsInfo')
> PTs.orh <- aperm(array(unlist(info),dim=c(length(info[[1]]),2,3)),
+                  c(1,3,2)
+                 )
> PRcurve(PTs.bp[,,1],PTs.bp[,,2],
+         main='PR curve',lty=1,xlim=c(0,1),ylim=c(0,1),
+         avg='vertical')
> PRcurve(PTs.lof[,,1],PTs.lof[,,2],
+         add=T,lty=2,
+         avg='vertical')
> PRcurve(PTs.orh[,,1],PTs.orh[,,2],
+         add=T,lty=1,col='grey',
+         avg='vertical')
> legend('topright',c('BPrule','LOF','ORh'),
+        lty=c(1,2,1),col=c('black','black','grey'))
> CRchart(PTs.bp[,,1],PTs.bp[,,2],
+         main='Cumulative Recall curve',lty=1,xlim=c(0,1),ylim=c(0,1),
+         avg='vertical')
> CRchart(PTs.lof[,,1],PTs.lof[,,2],
+         add=T,lty=2,
+         avg='vertical')
```

```
> CRchart(PTs.orh[,,1],PTs.orh[,,2],
+         add=T,lty=1,col='grey',
+         avg='vertical')
> legend('bottomright',c('BPrule','LOF','ORh'),
+        lty=c(1,2,1),col=c('black','black','grey'))
```

正如你可以看到，就累积回溯精确度而言，OR_h 方法的结果可与 LOF 方法相比。然而，对于 PR 曲线，OR_h 方法明显优于 LOF 方法，以较小的优势超过 BPrule 方法。

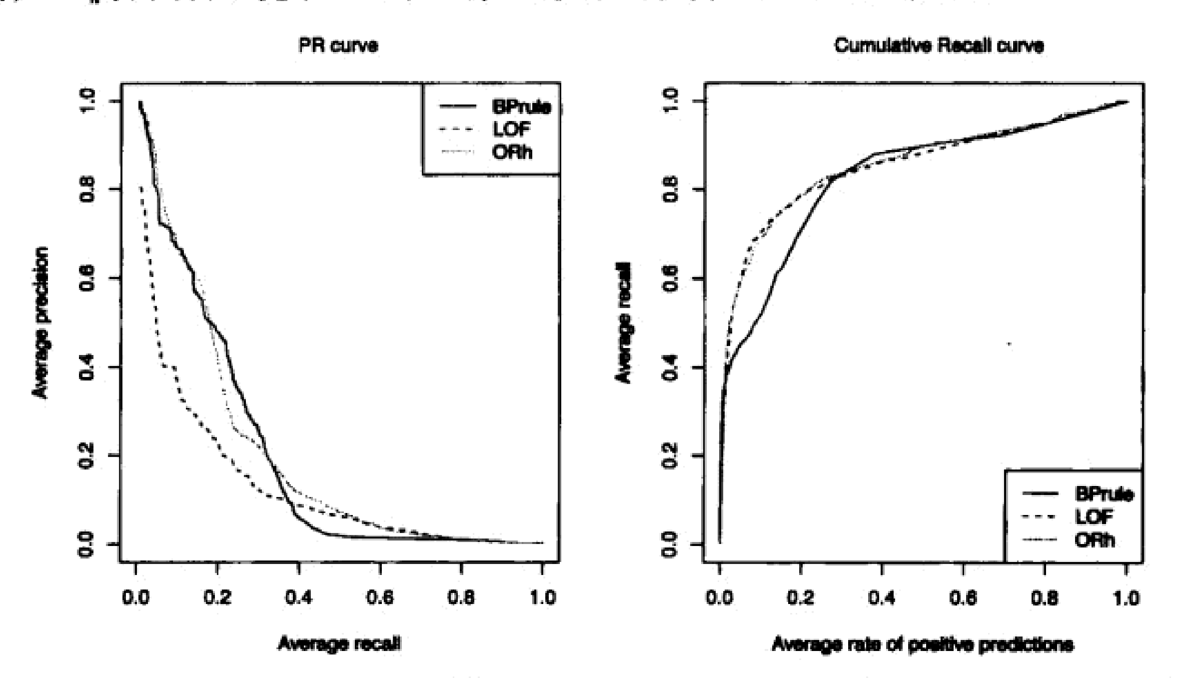


图4-9 OR_h、LOF 以及 BPrule 方法的 PR 图（左图）和累积回溯精确度曲线（右图）

4.4.2 有监督方法

在本节中我们探讨一些解决本章案例的有监督分类方法。考虑到我们的目标是获得一组交易报告的排序，我们将有约束地选择模型。我们将只使用能够产生概率分类的模型。对于每个测试案例，这些模型将输出它属于每一个可能类的概率。这些信息将使我们能够根据测试集案例属于“目标”类（即欺诈报告）的概率来对它们进行排序。

在描述我们将要用到的一些具体分类算法之前，先讨论数据集所特有的问题：类标签分布失衡。

4.4.2.1 类失衡问题

我们数据集的正常报告和欺诈报告的比例很不平衡。后者明显是少数，大约只占到检验报告（即有监督的个案）的8.1%。这种类型的问题可以导致预测模型建模过程中的各种困难。首先，它们需要合适的评价指标，因为众所周知标准的精确度（或者它的对立面，错误率）在这些领域是明显不够的。事实上，在我们的应用中，如果预测所有报告为正常报告将会很容易地获得90%左右的决策精确度。由于该类占大多数，所以它将给我们显得非常高的决策精确度水平。类失衡导致的另一个问题是，它会强烈地影响学习算法的性能，算法由于少数类缺乏统计支持而忽视它们。如果少数类恰恰是最相关的类，就像我们的案例一样，那么失衡导致的问题尤其严重。

有多种有助于学习算法克服类失衡问题的技术。它们一般分为两类：1）偏置学习过程的方法，它应用特定的对少数类更敏感的评价指标；2）用抽样方法来操作训练数据，从而改变类的分布。我们尝试使用的有监督分类方法属于第二类方法。

208
≀
209

有多种抽样方法用于改变数据集中类的失衡。一种抽样方法是欠采样法，它从多数类中选择一小部分个案，并把它们和少数类个案一起构成一个有更加平衡的类分布的数据集。另一种是过采样方法，它采用另外的工作模式，使用某些进程来复制少数类个案。有许多上述两种采样方法的变

体。一个成功的例子是 SMOTE 方法（Chawla et al.，2002）。这种方法的思路是用少数类个案的最近邻居来人工产生少数类的个案。此外，多数类的个案仍然采用欠抽样方法。这样就得到了一个更加平衡的数据集。本书添加包中的函数 SMOTE( ) 已经实现了这种抽样方法。基于失衡的样本，该函数生成一个有更加平衡类分布的新数据集。下面的代码给出了它的一个简单示例：

```
> data(iris)
> data <- iris[, c(1, 2, 5)]
> data$Species <- factor(ifelse(data$Species == "setosa", "rare",
+     "common"))
> newData <- SMOTE(Species ~ ., data, perc.over = 600)
> table(newData$Species)

common   rare
   600    350
```

这个小例子使用 iris 数据创建一个有两个预测变量（为易于可视化）和一个新的目标变量的人工数据集，它有失衡的类分布。该代码用参数 perc. over 的值 600 调用函数 SMOTE( )，这意味着将为初始数据集少数类的每个个案产生 6 个新样本。这些新样本采用对初始个案和其最近邻居（默认情况下为 5）的某种形式的随机插值方式产生。我们的实现采用混合模式的距离函数，所以可以在 SMOTE( ) 函数中应用同时含有连续变量和名义变量的数据集。

可以通过绘制原始数据集和 SMOTE 方法得到数据集，从而了解该方法。下面代码的结果如图 4-10 所示。

```
> par(mfrow = c(1, 2))
> plot(data[, 1], data[, 2], pch = 19 + as.integer(data[, 3]),
+     main = "Original Data")
> plot(newData[, 1], newData[, 2], pch = 19 + as.integer(newData[,
+     3]), main = "SMOTE'd Data")
```

210 在有监督分类算法的实验中，我们将应用 SMOTE 方法给出的平衡训练集来尝试不同方法的变体。

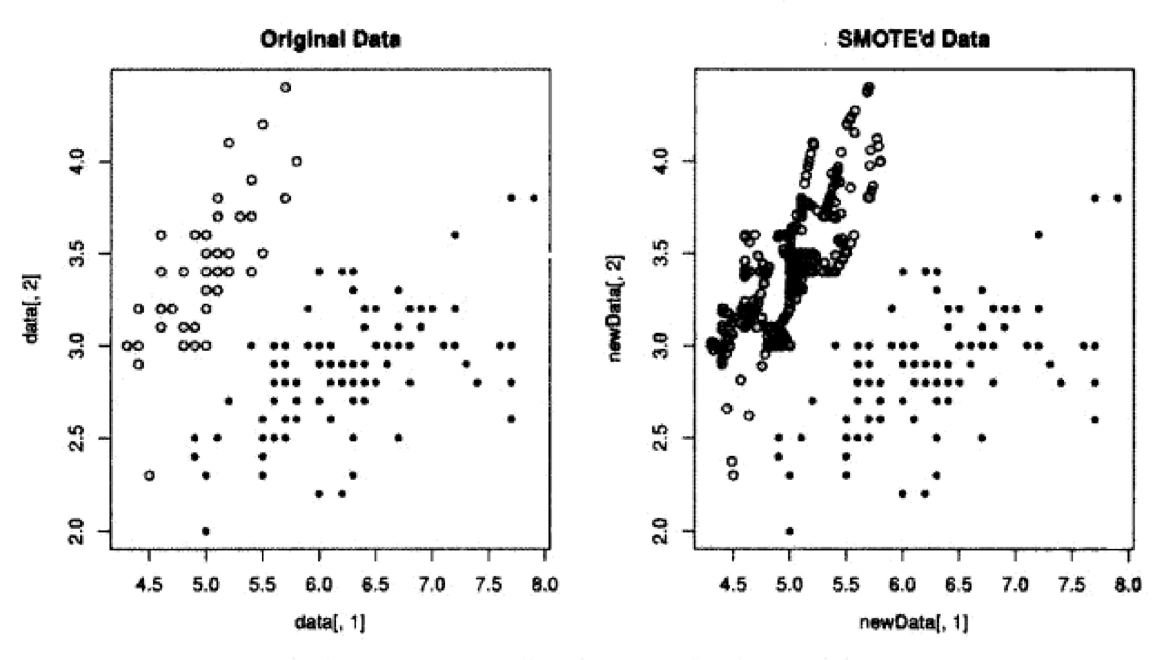


图 4-10 用 SMOTE 方法产生少数类的个案

类失衡的参考文献

类失衡是一个充分研究的课题。在几个这一特定主题的研讨会有这个课题研究的例子，如美国 AAAI'2000 和 ICML'2003 的失衡数据集研讨会，或 SIGKDD 的失衡数据集学习算法中的特殊问题（Chawla et al.，2004）。Chawla（2005）给出了一个很好的现有工作的概述。失衡类影响了预测模型的多个相关主题。例子包括预测模型的评价（例如，Provost 和 Fawcett（1997，2001）；

Provost et al. (1998)), 或成本敏感学习 (例如, Domingos (1999); Drummond 和 Holte (2006); Elkan (2001))。关于类失衡的采样方法, 有些参考文献包括 Kubat 和 Matwin (1997), Japkowicz (2000) 以及 Weiss 和 Provost (2003)。特别是 SMOTE 方法的主要参考文献是 Chawla et al. (2002) 和 Chawla et al. (2003)。

4.4.2.2 简单贝叶斯方法

简单贝叶斯 (Naive Bayes) 方法是基于贝叶斯定理的一种统计分类方法, 该理论应用到预测变量之间独立这一很强的假设条件。这些假设在实际问题中是很难成立的, 所以该方法名为“简单”。然而, 该方法还是很成功地应用于大量的实际问题。

贝叶斯定理假设 $P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$。简单贝叶斯分类应用这个定理来计算给定测试集个案条件下的类概率:

$$P(c \mid X_1, \cdots, X_p) = \frac{P(c)P(X_1, \cdots, X_p \mid c)}{P(X_1, \cdots, X_p)} \tag{4-5}$$

这里 c 是一个类, X_1, …, X_p 是给定测试集的预测变量的观测值。

211

概率 $P(c)$ 可以视为类 c 的先验预期, 而 $P(X_1, \cdots, X_p \mid c)$ 则是在给定类 c 的条件下测试个案的似然概率。最后, 分母是观测到的证据的概率。由于分母对所有类都是常数, 所以决策仅仅取决于方程中的分子。应用条件概率的统计定义和预测变量间独立的条件假设 (简单的), 推导出分式中的分子为:

$$P(c)P(X_1, \cdots, X_p \mid c) = P(c)\prod_{i=1}^{p} P(X_i \mid c) \tag{4-6}$$

简单贝叶斯方法用训练集样本的相对频率来估计这些概率。应用这些估计值, 该方法按照式(4-5) 来输出每一个测试个案的类概率。

R 有多个简单贝叶斯方法的实现。我们将应用添加包 e1071 中的函数 naiveBayes()。添加包 klaR (Weihs et al., 2005) 也包含了贝叶斯分类的实现, 同时还提供了有趣的可视化函数。

下面的函数用简单贝叶斯方法得到测试集报告的离群值排序分数。它应用给定的训练集样本中检验过的报告来得到简单贝叶斯分类模型。通过估计属于类 fraud 的概率来得到离群值排序:

```
> nb <- function(train, test) {
+     require(e1071, quietly = T)
+     sup <- which(train$Insp != "unkn")
+     data <- train[sup, c("ID", "Prod", "Uprice", "Insp")]
+     data$Insp <- factor(data$Insp, levels = c("ok", "fraud"))
+     model <- naiveBayes(Insp ~ ., data)
+     preds <- predict(model, test[, c("ID", "Prod", "Uprice",
+         "Insp")], type = "raw")
+     return(list(rankOrder = order(preds[, "fraud"], decreasing = T),
+         rankScore = preds[, "fraud"]))
+ }
```

下面的函数将被保留 (hold-out) 例程调用, 得到简单贝叶斯预测的选定的统计评价指标:

```
> ho.nb <- function(form, train, test, ...) {
+   res <- nb(train,test)
+   structure(evalOutlierRanking(test,res$rankOrder,...),
+             itInfo=list(preds=res$rankScore,
+                         trues=ifelse(test$Insp=='fraud',1,0)
+                        )
+            )
+ }
```

最后, 调用函数 holdOut(), 采用和前面无监督模型相同的设置来对这个模型进行实验:

```
> nb.res <- holdOut(learner('ho.nb',
+                           pars=list(Threshold=0.1,
+                                     statsProds=globalStats)),
+                   dataset(Insp ~ .,sales),
+                   hldSettings(3,0.3,1234,T),
+                   itsInfo=TRUE
+                   )
```

对于 10% 的检验努力，简单贝叶斯模型的结果如下：

```
> summary(nb.res)

== Summary of a Hold Out Experiment ==
 Stratified  3 x 70 %/ 30 % Holdout run with seed =  1234

* Dataset :: sales
* Learner  ::  ho.nb  with parameters:
         Threshold  =  0.1
         statsProds  =  11.34  ...

* Summary of Experiment Results:

          Precision      Recall    avgNDTP
avg      0.013715365 0.43112103 0.8519657
std      0.001083859 0.02613164 0.2406771
min      0.012660336 0.40533333 0.5908980
max      0.014825920 0.45758355 1.0650114
invalid 0.000000000 0.00000000 0.0000000
```

得到的分数大大低于前面用无监督方法得到的最优分数。

下面我们绘制和前面一样的曲线，以得到该模型性能的整体视图。我们把简单贝叶斯方法和最好的无监督方法，OR_h，进行比较：

```
> par(mfrow=c(1,2))
> info <- attr(nb.res,'itsInfo')
> PTs.nb <- aperm(array(unlist(info),dim=c(length(info[[1]]),2,3)),
+                 c(1,3,2)
+                )
> PRcurve(PTs.nb[,,1],PTs.nb[,,2],
+         main='PR curve',lty=1,xlim=c(0,1),ylim=c(0,1),
+         avg='vertical')
> PRcurve(PTs.orh[,,1],PTs.orh[,,2],
+         add=T,lty=1,col='grey',
+         avg='vertical')
> legend('topright',c('NaiveBayes','ORh'),
+        lty=1,col=c('black','grey'))
> CRchart(PTs.nb[,,1],PTs.nb[,,2],
+         main='Cumulative Recall curve',lty=1,xlim=c(0,1),ylim=c(0,1),
+         avg='vertical')
> CRchart(PTs.orh[,,1],PTs.orh[,,2],
+         add=T,lty=1,col='grey',
+         avg='vertical')
> legend('bottomright',c('NaiveBayes','ORh'),
+        lty=1,col=c('black','grey'))
```

图 4-11 所示的图形表明简单贝叶斯方法明显不如 OR_h 方法好。两个图形都说明后者在所有的情况下都优于前者。

导致简单贝叶斯方法性能差的一个可能原因是问题的类失衡。4.4.2.1 节给出了几个解决类失衡问题的方法，特别是 SMOTE 算法。下面通过 SMOTE 来获得一个训练集，然后再应用贝叶斯分类方法。

212 ~ 214

下面的函数和之前函数的主要不同之处在于下面的函数对函数 naiveBayes( ) 的调用，这里用了一个修改过的训练集：

```
> nb.s <- function(train, test) {
+     require(e1071, quietly = T)
+     sup <- which(train$Insp != "unkn")
+     data <- train[sup, c("ID", "Prod", "Uprice", "Insp")]
+     data$Insp <- factor(data$Insp, levels = c("ok", "fraud"))
+     newData <- SMOTE(Insp ~ ., data, perc.over = 700)
+     model <- naiveBayes(Insp ~ ., newData)
+     preds <- predict(model, test[, c("ID", "Prod", "Uprice",
+         "Insp")], type = "raw")
+     return(list(rankOrder = order(preds[, "fraud"], decreasing = T),
+         rankScore = preds[, "fraud"]))
+ }
```

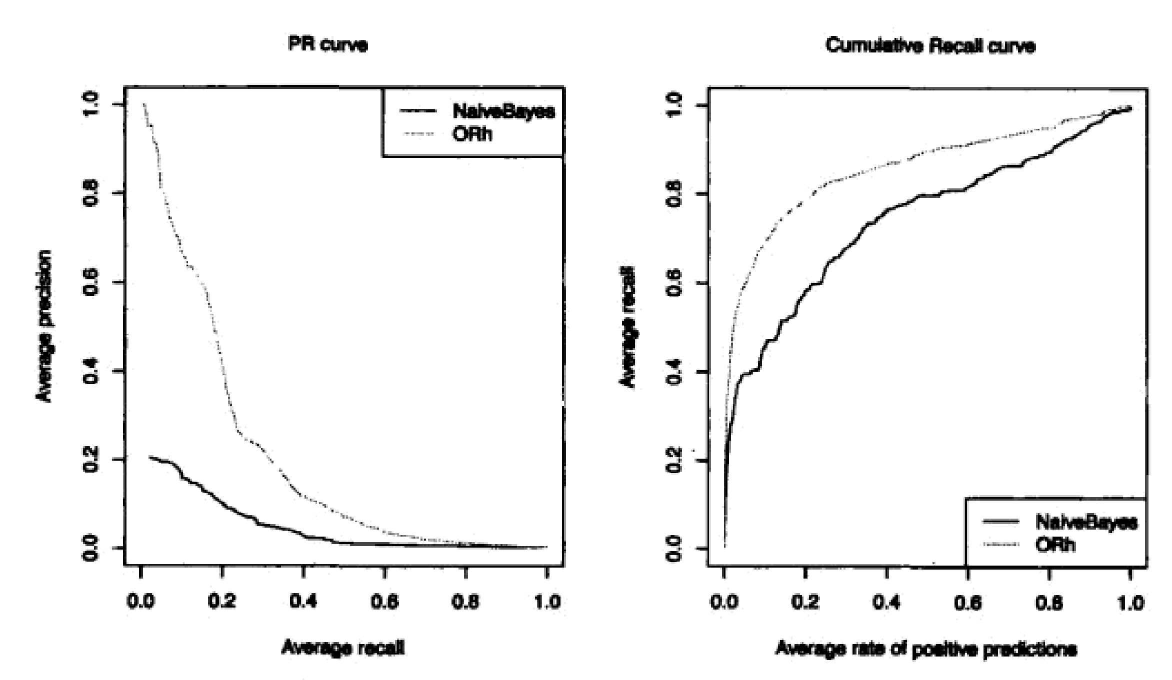


图 4-11　简单贝叶斯方法和 OR_h 方法的 PR 图（左图）和累积回溯精确度曲线（右图）

下面的语句获得对简单贝叶斯 SOMTE 版本模型的保留（hold-out）估计：

```
> ho.nbs <- function(form, train, test, ...) {
+   res <- nb.s(train,test)
+   structure(evalOutlierRanking(test,res$rankOrder,...),
+             itInfo=list(preds=res$rankScore,
+                         trues=ifelse(test$Insp=='fraud',1,0)
+                        )
+            )
+ }

> nbs.res <- holdOut(learner('ho.nbs',
+                            pars=list(Threshold=0.1,
+                                      statsProds=globalStats)),
+                    dataset(Insp ~ .,sales),
+                    hldSettings(3,0.3,1234,T),
+                    itsInfo=TRUE
+                    )
```

对于 10% 的检验努力，这个版本的简单贝叶斯模型的结果如下：

```
> summary(nbs.res)

== Summary of a Hold Out Experiment ==

 Stratified  3 x 70 %/ 30 % Holdout run with seed =  1234

* Dataset :: sales
* Learner  ::  ho.nbs  with parameters:
         Threshold  =  0.1
         statsProds  =  11.34  ...
* Summary of Experiment Results:

          Precision      Recall    avgNDTP
avg     0.014215115 0.44686510 0.8913330
std     0.001109167 0.02710388 0.8482740
min     0.013493253 0.43044619 0.1934613
max     0.015492254 0.47814910 1.8354999
invalid 0.000000000 0.00000000 0.0000000
```

这些结果和“正常”的简单贝叶斯方法几乎相差不大。得分仅仅稍微好一些，仍然和无监督模型的最好结果相差很远。它看起来尽管用 SMOTE 方法对少数类进行了过采样，但简单贝叶斯方法仍然不能正确预测哪些报告是有欺诈的。和前面一样，下面绘制这个方法的两个曲线图来得到该方法全局性能的全局视角。

```
> par(mfrow=c(1,2))
> info <- attr(nbs.res,'itsInfo')
> PTs.nbs <- aperm(array(unlist(info),dim=c(length(info[[1]]),2,3)),
+                  c(1,3,2)
+                 )
> PRcurve(PTs.nb[,,1],PTs.nb[,,2],
+         main='PR curve',lty=1,xlim=c(0,1),ylim=c(0,1),
+         avg='vertical')
> PRcurve(PTs.nbs[,,1],PTs.nbs[,,2],
+         add=T,lty=2,
+         avg='vertical')
> PRcurve(PTs.orh[,,1],PTs.orh[,,2],
+         add=T,lty=1,col='grey',
+         avg='vertical')
> legend('topright',c('NaiveBayes','smoteNaiveBayes','ORh'),
+        lty=c(1,2,1),col=c('black','black','grey'))
> CRchart(PTs.nb[,,1],PTs.nb[,,2],
+         main='Cumulative Recall curve',lty=1,xlim=c(0,1),ylim=c(0,1),
+         avg='vertical')
> CRchart(PTs.nbs[,,1],PTs.nbs[,,2],
+         add=T,lty=2,
+         avg='vertical')
> CRchart(PTs.orh[,,1],PTs.orh[,,2],
+         add=T,lty=1,col='grey',
+         avg='vertical')
> legend('bottomright',c('NaiveBayes','smoteNaiveBayes','ORh'),
+        lty=c(1,2,1),col=c('black','black','grey'))
```

图 4-12 确认了这个 SMOTE 版本的简单贝叶斯方法令人失望的结果。事实上，它说明与 OR_h 方法相比，它和“标准”简单贝叶斯方法有同样差的结果，而且它的性能总是不如标准的贝叶斯方法好。

215
~
216

基于这些结果，读者可能问，这里为什么没有和前面的无监督方法一样，按照独立的产品来建立模型，也许这就是问题所在呢。作为练习，你可以按照这种思路来运行简单贝叶斯模型。你需要做的就是按照前面无监督模型中代码所做的，按照产品来分割交易，然后应用简单贝叶斯模型。如果你进行这个练习，你将遇到的一个额外困难是有太少的产品监督报告。事实上，即使没有有标记这一限制，我们还是可以看到多个产品有太少的交易。如果加上了有标记的交易这一限制条件，这个问题肯定会更严重。

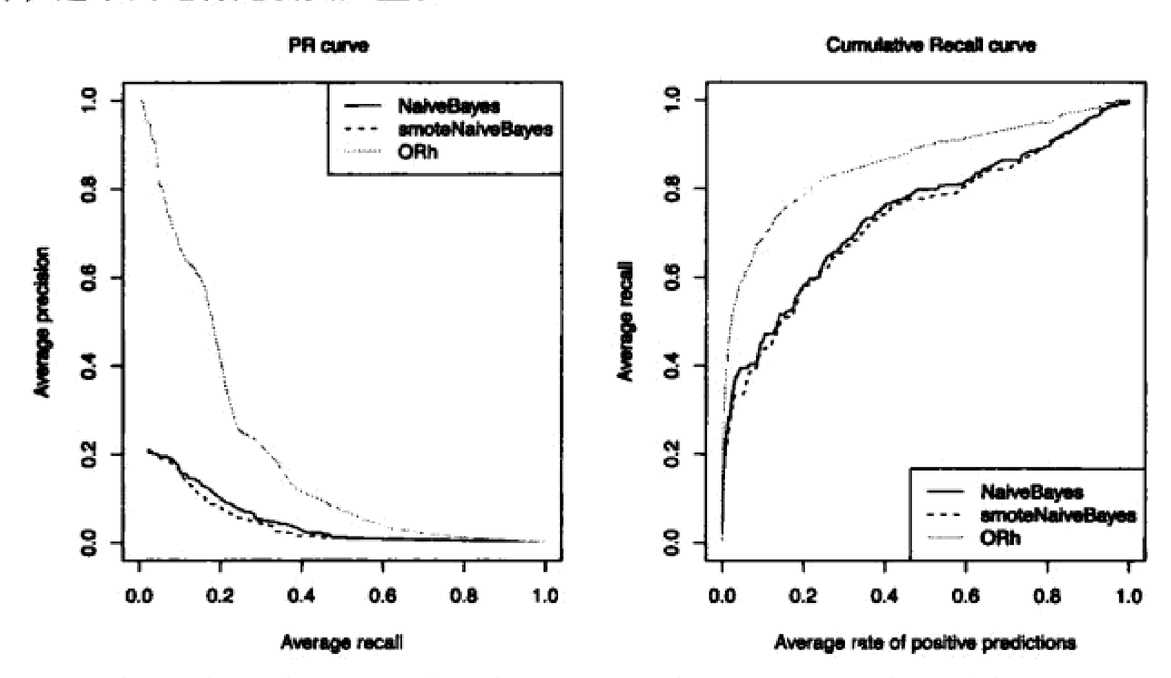


图4-12　两个版本的简单贝叶斯方法和 OR_h 方法的PR图（左图）和累积回溯精确度曲线（右图）

简单贝叶斯方法的参考文献

简单贝叶斯方法是许多研究领域很有名的分类算法。该主题的其他参考资料包括 Domingos 和 Pazzani（1997）的文章，Rish（2001）的文章，Hand 和 Yu（2001）的文章以及 Kononenko（1991）的文章。

4.4.2.3　AdaBoost 方法

AdaBoost 方法（Freund and Shapire，1996）是属于组合方法的一种学习算法。事实上，对于这种类型的算法，它们的预测值是通过对一组基本模型的预测值进行某种形式的组合而形成的。AdaBoost 方法应用一种自适应增强方法来得到一组基本模型。假如它比随机分类器好，那么增强方法是一种常见的提高基本算法性能的方法。AdaBoost 模型是通过序贯方式来获取的。序列的每一个新成员都是通过提高序列中前一个模型的误差率来获得的。它通过一种加权模式来提高模型性能：它增加被前一个模型误分类的个案的权重。这意味着基本模型用于不同分布的训练集数据。经过以上过程的几次迭代，结果是一组在不同训练集数据上的基本模型。这个组合可以用于获得原始数据的测试个案的预测值。对单个基本模型的预测值进行加权平均就可以得到组合预测值。权重的定义方式是，最大的权重赋给序列中最后得到的模型（理论上最小误差的模型）。 217

AdaBoost 应用的加权方式对于类分布失衡的学习算法很有意义。即使在初始的迭代中，也有少数个案的类被模型忽略，它们的权重将会增加，模型“被迫”学习它们。理论上，这将导致得到的组合模型能更精确地预测这些稀有个案。

AdaBoost. M1 是 AdaBoost 方法的一个特殊实现。它用具有少数结点的回归树作为基本模型。添加包 adabag（Cortes et al.，2010）中的函数 adaboost. M1( ) 实现了该方法。不幸的是，该模型提供的 predict 方法不能给出类概率。对我们的案例而言，这是一个严重的限制。如前所述，我们需要这些概率值，因为我们要用每个报告属于类 fraud 的概率来得到离群值排序。因此，我们

这里不能应用这个 AdaBoost. M1 算法的实现。到本书写作时，上面的函数是唯一的 AdaBoost. M1 算法的 R 实现。然而，我们可以选择 Weka [⊖]数据挖掘软件。Weka 是一个数据挖掘和机器学习的开源软件。这款优秀的软件提供了很多具有友好用户界面的学习算法。与 R 相比，它提供了 R 中所没有的多个算法，同时它有易于应用的漂亮用户界面。另一方面，R 在软件开发、开发协议方面有更好的灵活性，并且它有适用于更多研究领域的建模工具。由于 R 的添加包 RWeka (Hornik et al. , 2009)，我们才可以在 R 内应用 Weka 软件的大部分功能。如果计算机上安装有 Java 软件，那么在安装该添加包的同时也会在该计算机上安装 Weka 软件。如果没有安装 Java 软件，那么安装过程会报错并明确给出如何做的指示。我们强烈建议安装完该添加包后，要阅读它的帮助文档以了解 RWeka 可以应用哪些方法。

218

添加包 RWeka 的函数 AdaBoostM1( ) 通过 Weka 实现了 AdaBoost. M1 分类模型。与添加包 adabag 的实现相反，该算法的 predict 方法可以输出类概率，因此可以用它来得到我们问题中的离群值排序。默认情况下，Weka 的实现用决策桩为基本模型，决策桩是一类特殊的分类树，它仅有一个单独的测试结点。这个默认值和其他设置都是函数的参数，如有必要都可以修改。函数 WOW( ) 允许你检查一个特定的 Weka 算法有哪些参数。下面是该函数在我们的目标模型中应用的一个例子：

```
> library(RWeka)
> WOW(AdaBoostM1)

-P      Percentage of weight mass to base training on.  (default
        100, reduce to around 90 speed up)
        Number of arguments: 1.
-Q      Use resampling for boosting.
-S      Random number seed.  (default 1)
        Number of arguments: 1.
-I      Number of iterations.  (default 10)
        Number of arguments: 1.
-D      If set, classifier is run in debug mode and may output
        additional info to the console
-W      Full name of base classifier.  (default:
        weka.classifiers.trees.DecisionStump)
        Number of arguments: 1.
-
-D      If set, classifier is run in debug mode and may output
        additional info to the console
```

当调用相应的函数时，可以通过参数 control 和函数 Weka_control( ) 对一些参数值进行改变。下面是对著名的数据集 iris，应用函数 AdaBoostM1( ) 的一个示例：

```
> data(iris)
> idx <- sample(150,100)
> model <- AdaBoostM1(Species ~ .,iris[idx,],
+                     control=Weka_control(I=100))
> preds <- predict(model,iris[-idx,])
> head(preds)

[1] setosa setosa setosa setosa setosa setosa
Levels: setosa versicolor virginica

> table(preds,iris[-idx,'Species'])
```

⊖ http: //www. cs. waikato. ac. nz/ml/weka/.

```
preds        setosa versicolor virginica
  setosa         19          0         0
  versicolor      0         13         1
  virginica       0          2        15

> prob.preds <- predict(model,iris[-idx,],type='probability')
> head(prob.preds)

      setosa   versicolor    virginica
2  0.9999942 5.846673e-06 2.378153e-11
4  0.9999942 5.846673e-06 2.378153e-11
7  0.9999942 5.846673e-06 2.378153e-11
9  0.9999942 5.846673e-06 2.378153e-11
10 0.9999942 5.846673e-06 2.378153e-11
12 0.9999942 5.846673e-06 2.378153e-11
```

这个小例子也说明了如何用该模型获得分类概率。

对于离群值排序问题，我们现在准备了应用这类模型的必要函数。与简单贝叶斯模型一样，我们在所有交易上应用 AdaBoost. M1 方法，而不是单个产品。下面的函数得到给定训练集和测试集的排序报告：

```
> ab <- function(train,test) {
+   require(RWeka,quietly=T)
+   sup <- which(train$Insp != 'unkn')
+   data <- train[sup,c('ID','Prod','Uprice','Insp')]
+   data$Insp <- factor(data$Insp,levels=c('ok','fraud'))
+   model <- AdaBoostM1(Insp ~ .,data,
+                       control=Weka_control(I=100))
+   preds <- predict(model,test[,c('ID','Prod','Uprice','Insp')],
+                    type='probability')
+   return(list(rankOrder=order(preds[,'fraud'],decreasing=T),
+               rankScore=preds[,'fraud'])
+         )
+ }
```

在保留（hold-out）例程中调用的函数：

```
> ho.ab <- function(form, train, test, ...) {
+   res <- ab(train,test)
+   structure(evalOutlierRanking(test,res$rankOrder,...),
+             itInfo=list(preds=res$rankScore,
+                         trues=ifelse(test$Insp=='fraud',1,0)
+                        )
+            )
+ }
```

最后，给出运行保留（hold-out）实验的代码：

219
~
220

```
> ab.res <- holdOut(learner('ho.ab',
+                          pars=list(Threshold=0.1,
+                                    statsProds=globalStats)),
+                   dataset(Insp ~ .,sales),
+                   hldSettings(3,0.3,1234,T),
+                   itsInfo=TRUE
+                   )
```

对于 10% 的检验努力，AdaBoost 模型的结果如下：

```
> summary(ab.res)

== Summary of a Hold Out Experiment ==
```

```
Stratified  3 x 70 %/ 30 % Holdout run with seed =  1234

* Dataset ::  sales
* Learner  ::  ho.ab  with parameters:
         Threshold  =  0.1
         statsProds  =  11.34  ...

* Summary of Experiment Results:
          Precision      Recall    avgNDTP
avg     0.0220722972 0.69416565 1.5182034
std     0.0008695907 0.01576555 0.5238575
min     0.0214892554 0.68241470 0.9285285
max     0.0230717974 0.71208226 1.9298286
invalid 0.0000000000 0.00000000 0.0000000
```

这些结果属于目前得到的最好结果之一。事实上，与 LOF 和 OR_h 得到的最好结果相比，这里的结果仍然很不错。另外，我们注意到这个模型仅仅应用给定报告的很小部分（检验过的报告）来得到它们的排序。尽管如此，它达到了稳健的回溯精确度值69%和一个很好的平均 NDTP 值1.5。

下面的代码得到 PR 曲线和累积回溯精确度曲线：

```
> par(mfrow=c(1,2))
> info <- attr(ab.res,'itsInfo')
> PTs.ab <- aperm(array(unlist(info),dim=c(length(info[[1]]),2,3)),
+                 c(1,3,2)
+                 )
> PRcurve(PTs.nb[,,1],PTs.nb[,,2],
+         main='PR curve',lty=1,xlim=c(0,1),ylim=c(0,1),
+         avg='vertical')
> PRcurve(PTs.orh[,,1],PTs.orh[,,2],
+         add=T,lty=1,col='grey',
+         avg='vertical')
> PRcurve(PTs.ab[,,1],PTs.ab[,,2],
+         add=T,lty=2,
+         avg='vertical')
> legend('topright',c('NaiveBayes','ORh','AdaBoostM1'),
+        lty=c(1,1,2),col=c('black','grey','black'))
> CRchart(PTs.nb[,,1],PTs.nb[,,2],
+         main='Cumulative Recall curve',lty=1,xlim=c(0,1),ylim=c(0,1),
+         avg='vertical')
> CRchart(PTs.orh[,,1],PTs.orh[,,2],
+         add=T,lty=1,col='grey',
+         avg='vertical')
> CRchart(PTs.ab[,,1],PTs.ab[,,2],
+         add=T,lty=2,
+         avg='vertical')
> legend('bottomright',c('NaiveBayes','ORh','AdaBoostM1'),
+        lty=c(1,1,2),col=c('black','grey','black'))
```

图4-13也确认了 AdaBoost. M1 算法的优秀性能，特别是累积回溯精确度。该曲线显示对于大多数的资源限制水平，AdaBoost. M1 方法的分数和 OR_h 方法得到的分数匹配。就 PR 曲线而言，尤其是对低水平的回溯精确度值，AdaBoost. M1 方法的性能不是那么吸引人。然而，对于较高的回溯精确度值，它明显地与我们目前得到的最好的决策精确度匹配。而且，我们注意到，这里较高的回溯精确度水平恰恰是我们的应用所需要的。

221～222

总之，对我们的应用而言，AdaBoost. M1 方法是一个很有竞争力的模型。尽管存在类失衡问

题，但该组合方法给出了具有最好性能的产品排序。

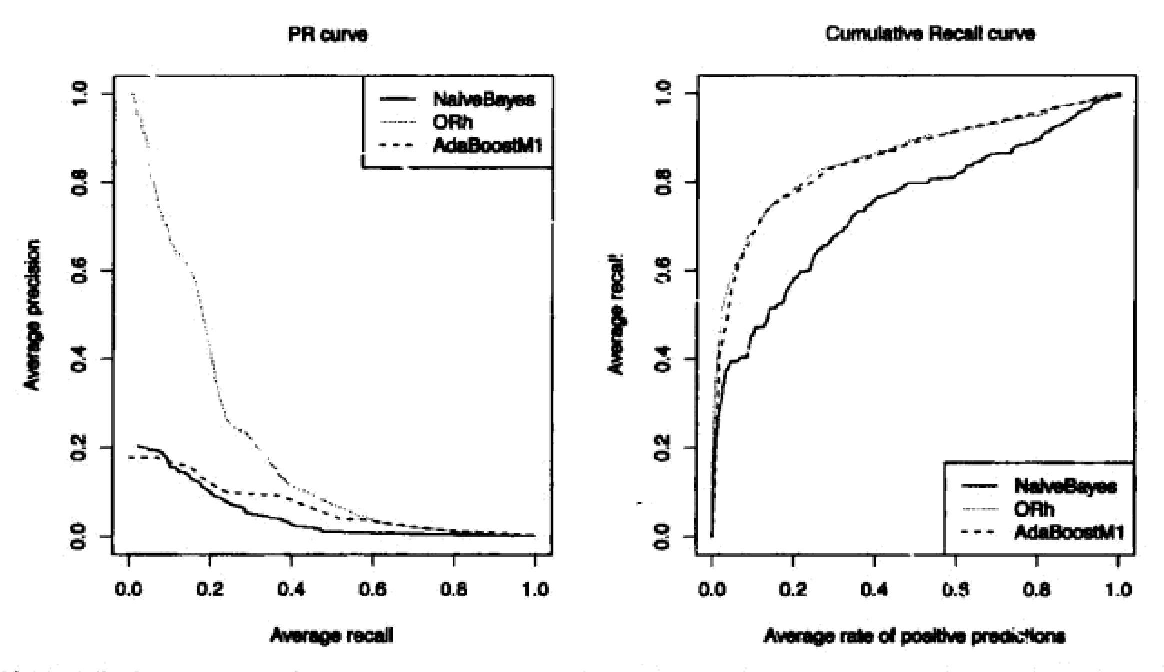


图 4-13 简单贝叶斯方法、OR_h 方法和 AdaBoost. M1 方法的 PR 图（左图）和累积回溯精确度曲线（右图）

增强方法的参考文献

AdaBoost. M1 算法是更广泛增强类算法的一个例子，该类算法应用较差学习算法（比随机猜测稍好）的组合来获得较好的预测性能。这类算法的参考资料是 Freund 和 Shapire（1996）的论文。其他重要的增强方法的历史文献是 Shapire（1990）和 Freund（1990）的文章。有些重要的分析可以在 Breiman（1998）、Friedman（2002）和 Rätsch 等（2001）的工作中找到。可以在 Hastie 等（2001）的书籍的第 10 章找到有关增强方法的很好描述。

4.4.3 半监督方法

本节描述尝试同时使用检验的和没有检验的报告来得到侦测欺诈报告的分类模型。这意味着我们需要某种形式的半监督分类模型（参见 4.3.1.3 节）。

自我训练模型（例如，Rosenberg 等（2005）；Yarowsky（1995））是一个众所周知的半监督分类形式。该方法先用给定标记的个案来建立一个初始的分类器。然后应用这个分类器来预测给定训练集中未标记的个案。将分类器中有较高置信度的预测标签所对应的个案和预测的标签一起加入到有标记的数据集中。在这个新的数据集上我们得到一个新的分类器，继续进行这个过程，直到达到某个收敛准则时迭代过程才停止。只要能输出预测的置信度信息，那么任何基本分类算法都可运用该方法。这与 4.4.3 节描述的两个分类器的类概率相似。自我训练方法有三个相关的参数：1）基本训练模型；2）分类置信度阈值，它用来确定哪些个案加入到新的训练集中；3）决定何时终止自我训练过程的收敛准则。在本书的 R 添加包中，有一个泛型函数（SelfTrain()），它可以用于概率分类器，基于同时有标记个案和未标记个案的训练集来训练模型。

下面用数据集 iris 给出这个函数的一个简单应用示例。我们在该数据集中人工创建了少数未标记的样本，这样就能应用半监督的分类方法：

```
> library(DMwR)
> library(e1071)
> data(iris)
> idx <- sample(150, 100)
> tr <- iris[idx, ]
> ts <- iris[-idx, ]
> nb <- naiveBayes(Species ~ ., tr)
> table(predict(nb, ts), ts$Species)
```

```
> pred.nb <- function(m,d) {
+   p <- predict(m,d,type='raw')
+   data.frame(cl=colnames(p)[apply(p,1,which.max)],
+              p=apply(p,1,max)
+              )
+ }
> nb.st <- function(train,test) {
+   require(e1071,quietly=T)
+   train <- train[,c('ID','Prod','Uprice','Insp')]
+   train[which(train$Insp == 'unkn'),'Insp'] <- NA
+   train$Insp <- factor(train$Insp,levels=c('ok','fraud'))
+   model <- SelfTrain(Insp ~ .,train,
+                      learner('naiveBayes',list()),'pred.nb')
+   preds <- predict(model,test[,c('ID','Prod','Uprice','Insp')],
+                    type='raw')
+   return(list(rankOrder=order(preds[,'fraud'],decreasing=T),
+               rankScore=preds[,'fraud'])
+         )
+ }
> ho.nb.st <- function(form, train, test, ...) {
+   res <- nb.st(train,test)
+   structure(evalOutlierRanking(test,res$rankOrder,...),
+             .itInfo=list(preds=res$rankScore,
+                          trues=ifelse(test$Insp=='fraud',1,0)
+                         )
+            )
+ }
> nb.st.res <- holdOut(learner('ho.nb.st',
+                              pars=list(Threshold=0.1,
+                                        statsProds=globalStats)),
+                      dataset(Insp ~ .,sales),
+                      hldSettings(3,0.3,1234,T),
+                      itsInfo=TRUE
+                      )
```

运行上面的自我训练模型的结果如下所示。

```
> summary(nb.st.res)

== Summary of a Hold Out Experiment ==

 Stratified  3 x 70 %/ 30 % Holdout run with seed =  1234

* Dataset ::  sales
* Learner  ::  ho.nb.st  with parameters:
         Threshold  =  0.1
         statsProds  =  11.34  ...

* Summary of Experiment Results:

          Precision     Recall     avgNDTP
avg      0.013521017 0.42513271 1.08220611
std      0.001346477 0.03895915 1.59726790
min      0.012077295 0.38666667 0.06717087
max      0.014742629 0.46456693 2.92334375
invalid 0.000000000 0.00000000 0.00000000
```

这些结果很令人失望。它与只用有标记数据训练的简单贝叶斯的结果很相似。除了 NDTP 的

```
              setosa versicolor virginica
  setosa          12          0         0
  versicolor       0         21         1
  virginica        0          0        16

> trST <- tr
> nas <- sample(100, 90)
> trST[nas, "Species"] <- NA
> func <- function(m, d) {
+     p <- predict(m, d, type = "raw")
+     data.frame(cl = colnames(p)[apply(p, 1, which.max)],
+         p = apply(p, 1, max))
+ }
> nbSTbase <- naiveBayes(Species ~ ., trST[-nas, ])
> table(predict(nbSTbase, ts), ts$Species)

              setosa versicolor virginica
  setosa          12          0         0
  versicolor       0         18         2
  virginica        0          3        15

> nbST <- SelfTrain(Species ~ ., trST, learner("naiveBayes",
+     list()), "func")
> table(predict(nbST, ts), ts$Species)

              setosa versicolor virginica
  setosa          12          0         0
  versicolor       0         20         2
  virginica        0          1        15
```

上面的代码得到了3个不同的简单贝叶斯模型。第一个模型（nb）用一个有100个标记个案的样本得到。将这100个个案转换为另一个集合，其中的90个个案的目标变量被设为NA，于是它们变为未标记的个案。用剩余的10个有标记的个案得到第二个简单贝叶斯模型（nbSTbase）。最后，把有标记个案和未标记个案混合后的数据集传递给函数SelfTrain（）从而得到第三个模型（nbST）。从上面可以看到，在这个小例子中，自我训练的模型几乎达到了与用100个标记个案得到的初始模型一样的性能。

为了应用函数SelfTrain()，用户必须创建一个函数（上面代码中是func()），它能接收一个给定模型与测试集，并返回一个含有两列并与测试集有相同行数的数据框。数据框的第一列包含个案的预测标记，第二列为那个分类的相应概率。该函数需要在函数SelfTrain（）之外来定义，因为不是所有的predict方法都用相同的语法来获得类概率。

223～224

函数SelfTrain（）有几个参数来控制迭代过程。参数thrConf（默认值为0.9）设置为把一个未标记个案合并到有标记个案集的概率阈值（默认为0.9），参数maxIts允许用户设置自我训练迭代的最大次数（默认为10），而参数percFull（默认为1）用来指示如果有标记集合达到了某个给定数据集的一定百分比就停止该过程。自我训练迭代过程达到下列条件将停止：要么没有分类达到所要求的概率水平，要么达到了最大迭代次数，要么当前有标记训练集达到了给定数据集的目标比例。最后要注意的是，函数SelfTrain（）要求在目标变量上用NA值来表示未标记的个案。我们采用简单贝叶斯模型来应用这种自我训练。下面的代码实现和运行用自我训练的贝叶斯模型来进行的保留（hold-out）实验。这里要警告的是，有关运行该实验所必需的计算资源。取决于计算机硬件，运行的下面的代码将花费较长的时间，尽管是以分钟计算（至少在我的平均水平的计算机上是这样）。

平均值有些许提高外，其他统计量基本相同，它比我们目前得到的最好分数相差很大。而且，就是这个较好的指标还伴随一个很大的标准差。

图 4-14 给出了这个模型、标准简单贝叶斯模型和 OR_h 模型的 PR 曲线以及累积回溯精确度曲线。用下面的代码来绘制该图形：

```
> par(mfrow=c(1,2))
> info <- attr(nb.st.res,'itsInfo')
> PTs.nb.st <- aperm(array(unlist(info),dim=c(length(info[[1]]),2,3)),
+                    c(1,3,2)
+                   )
> PRcurve(PTs.nb[,,1],PTs.nb[,,2],
+         main='PR curve',lty=1,xlim=c(0,1),ylim=c(0,1),
+         avg='vertical')
> PRcurve(PTs.orh[,,1],PTs.orh[,,2],
+         add=T,lty=1,col='grey',
+         avg='vertical')
> PRcurve(PTs.nb.st[,,1],PTs.nb.st[,,2],
+         add=T,lty=2,
+         avg='vertical')
> legend('topright',c('NaiveBayes','ORh','NaiveBayes-ST'),
+        lty=c(1,1,2),col=c('black','grey','black'))
> CRchart(PTs.nb[,,1],PTs.nb[,,2],
+         main='Cumulative Recall curve',lty=1,xlim=c(0,1),ylim=c(0,1),
+         avg='vertical')
> CRchart(PTs.orh[,,1],PTs.orh[,,2],
+         add=T,lty=1,col='grey',
+         avg='vertical')
> CRchart(PTs.nb.st[,,1],PTs.nb.st[,,2],
+         add=T,lty=2,
+         avg='vertical')
> legend('bottomright',c('NaiveBayes','ORh','NaiveBayes-ST'),
+        lty=c(1,1,2),col=c('black','grey','black'))
```

图 4-14 确认了自我训练的简单贝叶斯分类模型令人失望的性能。对于这个特定问题，即使应用了由相对较小的数据集得到的简单贝叶斯模型，这个半监督分类器还是明显没有竞争力。

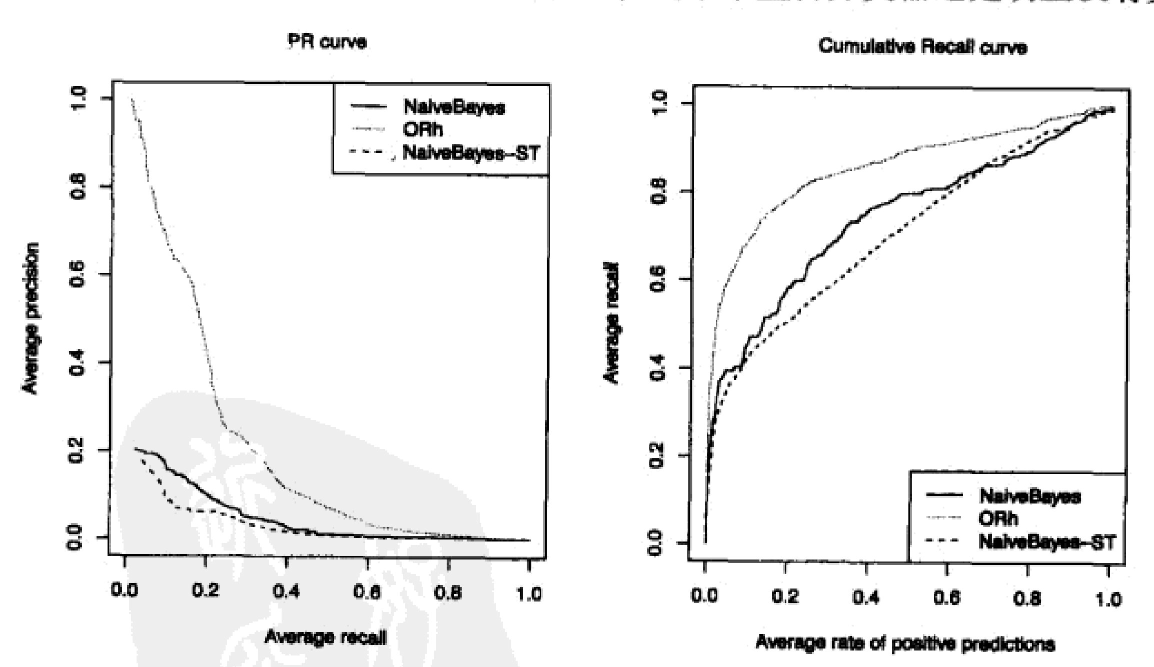


图 4-14　自训练的简单贝叶斯方法、标准简单贝叶斯方法和 OR_h 方法的 PR 图（左图）和累积回溯精确度曲线（右图）

我们也用 AdaBoost. M1 算法进行了自我训练。下面的代码执行这些实验：

225
~
227

```
> pred.ada <- function(m,d) {
+   p <- predict(m,d,type='probability')
+   data.frame(cl=colnames(p)[apply(p,1,which.max)],
+              p=apply(p,1,max)
+              )
+ }
> ab.st <- function(train,test) {
+   require(RWeka,quietly=T)
+   train <- train[,c('ID','Prod','Uprice','Insp')]
+   train[which(train$Insp == 'unkn'),'Insp'] <- NA
+   train$Insp <- factor(train$Insp,levels=c('ok','fraud'))
+   model <- SelfTrain(Insp ~ .,train,
+                      learner('AdaBoostM1',
+                              list(control=Weka_control(I=100))),
+                      'pred.ada')
+   preds <- predict(model,test[,c('ID','Prod','Uprice','Insp')],
+                    type='probability')
+   return(list(rankOrder=order(preds[,'fraud'],decreasing=T),
+               rankScore=preds[,'fraud'])
+          )
+ }
> ho.ab.st <- function(form, train, test, ...) {
+   res <- ab.st(train,test)
+   structure(evalOutlierRanking(test,res$rankOrder,...),
+             itInfo=list(preds=res$rankScore,
+                         trues=ifelse(test$Insp=='fraud',1,0)
+                         )
+             )
+ }
> ab.st.res <- holdOut(learner('ho.ab.st',
+                              pars=list(Threshold=0.1,
+                                        statsProds=globalStats)),
+                      dataset(Insp ~ .,sales),
+                      hldSettings(3,0.3,1234,T),
+                      itsInfo=TRUE
+                      )
```

在 10% 的检验努力下，自我训练的 AdaBoost 方法的结果如下：

```
> summary(ab.st.res)

== Summary of a Hold Out Experiment ==

 Stratified  3 x 70 %/ 30 % Holdout run with seed =  1234

* Dataset :: sales
* Learner  ::  ho.ab.st  with parameters:
         Threshold  =  0.1
         statsProds  =  11.34  ...
* Summary of Experiment Results:

          Precision      Recall    avgNDTP
avg     0.022377700 0.70365350 1.6552619
std     0.001130846 0.02255686 1.5556444
min     0.021322672 0.68266667 0.5070082
max     0.023571548 0.72750643 3.4257016
invalid 0.000000000 0.00000000 0.0000000
```

尽管不是出人意料地好，但这些分数比单独用有标记数据得到的 AdaBoost. M1 模型有提高。决策精确度基本上一样，在回溯精确度和平均 NDTP 上有小的提高。对于 10% 的努力水平，这里的回溯精确度是我们尝试的所有模型中最高的。图 4-15 给出了这个模型、标准 Adaboost. M1 模型和 OR_h 模型的 PR 曲线以及累积回溯精确度曲线。用下面的代码来绘制该图形：

```
> par(mfrow = c(1, 2))
> info <- attr(ab.st.res, "itsInfo")
> PTs.ab.st <- aperm(array(unlist(info), dim = c(length(info[[1]]),
+     2, 3)), c(1, 3, 2))
> PRcurve(PTs.ab[, , 1], PTs.ab[, , 2], main = "PR curve",
+     lty = 1, xlim = c(0, 1), ylim = c(0, 1), avg = "vertical")
> PRcurve(PTs.orh[, , 1], PTs.orh[, , 2], add = T, lty = 1,
+     col = "grey", avg = "vertical")
> PRcurve(PTs.ab.st[, , 1], PTs.ab.st[, , 2], add = T, lty = 2,
+     avg = "vertical")
> legend("topright", c("AdaBoostM1", "ORh", "AdaBoostM1-ST"),
+     lty = c(1, 1, 2), col = c("black", "grey", "black"))
> CRchart(PTs.ab[, , 1], PTs.ab[, , 2], main = "Cumulative Recall curve",
+     lty = 1, xlim = c(0, 1), ylim = c(0, 1), avg = "vertical")
> CRchart(PTs.orh[, , 1], PTs.orh[, , 2], add = T, lty = 1,
+     col = "grey", avg = "vertical")
> CRchart(PTs.ab.st[, , 1], PTs.ab.st[, , 2], add = T, lty = 2,
+     avg = "vertical")
> legend("bottomright", c("AdaBoostM1", "ORh", "AdaBoostM1-ST"),
+     lty = c(1, 1, 2), col = c("black", "grey", "black"))
```

在欺诈侦测问题所尝试的所有模型中，累积回溯精确度曲线确认了自我训练的 AdaBoost. M1 模型是最好的模型。特别是，在检验限值水平在 15% ~20% 时，就侦测出的欺诈报告的比例而言，它明显地好于其他的模型系统。就决策精确度而言，这个模型的分数不是太吸引人，但前面我们提到，那些被模型放在较高排序位置的未标记报告最终被确认为欺诈也不是一件坏事。

228 ~ 229

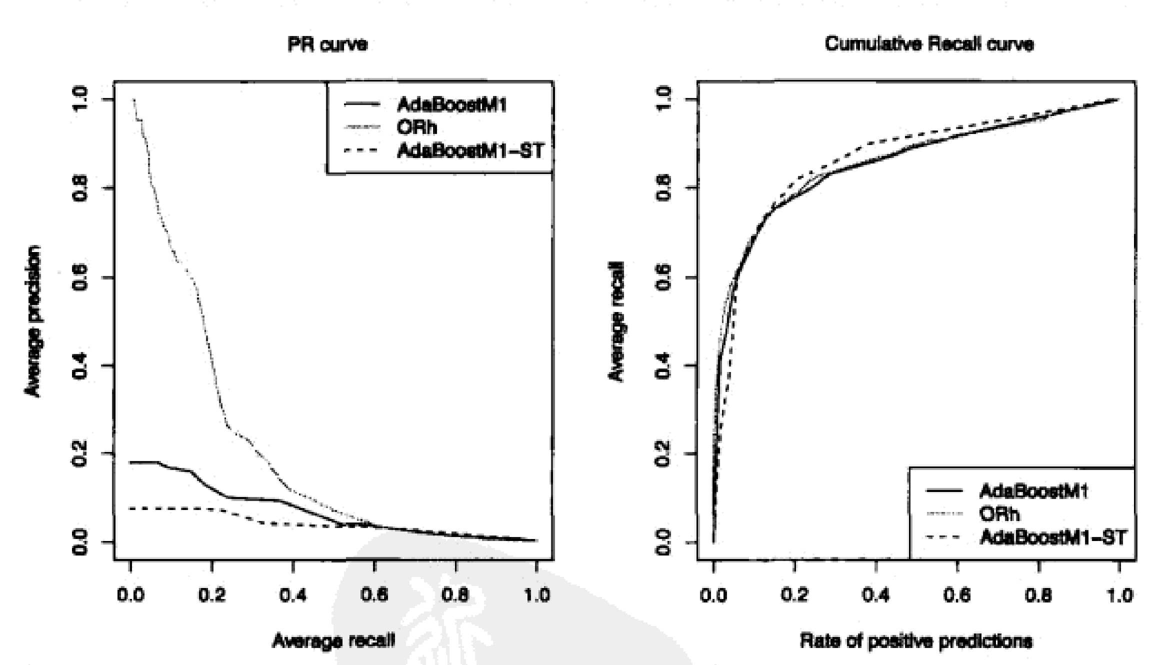


图 4-15　自训练的 AdaBoost. M1 方法、OR_h 方法和标准的 AdaBoost. M1 方法的 PR 图（左图）和累积回溯精确度曲线（右图）

4.5　小结

本章的主要目的是向读者介绍一类新的数据挖掘问题：离群值排序。特别是我们采用了一个从不同的方面处理这个任务的数据集。即，我们对该问题应用了有监督的、无监督的和半监督

的方法。对于应用有限的资源来找到一个现象的异常观测值这一常见问题，本章的应用可以看做是它的一个实例。多个实际问题可以映射到这个通用框架中，例如信用卡、电信和税收的欺诈侦测等。在证券领域，也有这种欺诈侦测一般概念的多个应用。

用方法论的术语来讲，我们介绍了下面的新主题：

- 离群值侦测和排序。
- 聚类方法。
- 半监督学习。
- 通过自我训练的半监督分类。
- 失衡的类分布以及处理这类问题的方法。 230
- 简单贝叶斯分类器。
- AdaBoost 分类器。
- 决策精确度/回溯精确度和累积回溯精确度。
- 保留（交叉验证）实验。

从学习 R 的角度来讲，我们演示了：

- 如何获取多个性能指标统计量和如何用 ROCR 添加包来可视化它们。
- 如何得到性能指标统计量的交叉验证估计。
- 如何用 LOF 方法得到局部离群值因子。
- 如何用 OR_h 方法得到离群值排序。
- 如何用 SMOTE 方法克服类失衡。
- 如何得到简单贝叶斯分类模型。
- 如何得到 AdaBoost. M1 分类模型。
- 如何通过 RWeka 添加包来应用 Weka 数据挖掘系统的方法。
- 如何用自训练方法把一个分类器应用到半监督的数据集。 231



微阵列样本分类

第四个案例是来自生物信息领域。在这个案例中，我们重点讲述如何对微阵列样本进行分类。更准确地说，给定一个描述病人基因表达情况的微阵列表，我们将确定这个病人的急性淋巴细胞白血病的基因突变类型。这个案例将涉及多个新的数据挖掘主题。根据这类数据的特点，本章重点讲述特征选择方法，也就是如何降低描述每个观察对象特征的个数。在这个特殊的应用中，我们给出几个常用的特征选择方法。本章涉及的其他数据挖掘主题包括 k 近邻分类，弃一交叉验证法和其他形式的组合模型。

5.1 问题描述与目标

生物信息学是 R 的主要应用领域之一。有一个基于 R 的生物信息学相关项目，该项目的目的是为生物信息学领域提供大量分析工具，该项目的名称为 Bioconductor[⊖]。本案例将使用这个项目提供的工具来处理有监督的分类问题。

5.1.1 微阵列实验背景简介

一个没有生物背景的人在生物信息学领域中碰到的主要问题之一是此领域中大量的“新”术语。在这个简单的背景介绍中，我们将向读者介绍生物信息学领域中的一些“术语”，并且试着把它们映射到更“标准”的数据挖掘术语。

分析差异基因表达是 DNA 微阵列实验中的一个重要应用之一。基因表达微阵列可以让我们根据样本的基因表达水平来描述这些样本（即个体）的特征。在生物信息学领域，一个样本是指某些研究现象的一个观测值（或者个案）。微阵列实验是用来测量这些观测值的“变量”集合的方法。这里的变量是指大量基因数据的集合，对每一个变量（基因），这些实验都会测量一个相应的表达水平。总之，一个数据集是由一组样本（即个案）构成，我们测量这些样本的大量基因（即变量）集合的表达水平。如果这些样品有一些与其相关的疾病状态，我们可以尝试找到一个未知函数，该函数可以把基因表达水平映射到疾病状态。可以使用一个先前分析过的样本的数据集来近似描述这个函数，这是一个有监督分类任务的一个实例，其中目标变量是疾病类型。在这种问题中，观测值是样品（微阵列、个体），预测变量是我们用微阵列实验来测量某值（即表达水平）的基因。这里的关键假设是不同的疾病类型是与不同的基因表达谱特征相关的，而且通过微阵列来测量这些基因表达谱，我们就可以精确地预测一个个体的疾病类型。

为了获得一些样本的基因表达水平，多种技术手段由此而生。短寡核苷酸芯片就是这些技术的一个例子。短寡核苷酸芯片的输出是一个图像，经过几个预处理步骤后，这个图像可以映射为一组基因表达水平。项目 Bioconductor 有几个 R 添加包就是实现这些预处理步骤的，它包括分析短寡核苷酸芯片输出的图像、标准化任务，以及其他几个得到基因表达水平的必要步骤。在这个案例中，我们不介绍这些初始的步骤。感兴趣的读者可以参考 Bioconductor 项目的几个信息资

233

⊖ http：//www.bioconductor.org.

源，也可以参考其他书籍（例如，Hahne et al.，2008）。

在这种情况下，我们将直接从经过这些预处理步骤得到的基因表达水平矩阵入手。该矩阵是观测值的预测变量信息。我们将会看到，通常会有比样本量更多的预测变量被测量，也就是说预测变量的个数多于观测值的个数。这是典型的微阵列数据集的特点。这些基因表达矩阵的另一个特别之处是，与标准的数据集相比较，它们看起来是经过转置的。这意味着矩阵的行代表预测变量（即基因），矩阵的列代表观测值（即样本）。对于每一个样本，我们也需要它的相关分类。在本章的案例中，该分类是一种遗传变异相关疾病。这里也可能存在其他的变量信息（例如，抽样个体的性别和年龄等）。

5.1.2 数据集 ALL

本章所使用的数据集来自关于急性淋巴细胞白血病的一个研究（Chiaretti et al.，2004；Li，2009）。这些数据是从患这类疾病的128位个体上得到的微阵列样本。事实上，在这些样本中有两种不同类型的肿瘤：T细胞ALL（共33例样本）和B细胞ALL（共95例样本）。 234

我们将集中研究B细胞ALL样本数据。在这后一组样品中，我们能区分出不同类型的突变，即ALL1/AF4、BCR/ABL、E2A/PBX1、p15/p16和没有细胞遗传学异常。在对B细胞ALL样本的分析中，由于只有一个样本的突变类型为p15/p16，所以所有分析中将剔除该观测值。我们建模的目标是根据它的微阵列能够预测出个体的突变类型。考虑到目标变量是有4个可能取值的名义变量，因此我们面临的是一个有监督的分类任务。

5.2 可用的数据

数据集ALL是生物信息学软件包（Bioconductor）的一部分。为了使用该数据集，我们至少需要从Bioconductor网站安装一些基本的添加包。由于这个数据集已经是R添加包的一部分，所以这里没有把这个数据集包含在本书的添加包中。

为了安装基本的生物信息学软件包和数据集ALL，假设我们已经正常连接到了互联网，需要执行下列指令：

```
> source("http://bioconductor.org/biocLite.R")
> biocLite()
> biocLite("ALL")
```

只需要在第一次使用时运行上面的代码。当把这些添加包安装好后，只要简单地运行下列代码就可以应用数据集ALL了：

```
> library(Biobase)
> library(ALL)
> data(ALL)
```

上面的指令会载入添加包Biobase（Gentleman et al.，2004）和添加包ALL（Gentleman et al.，2010）。然后，我们载入数据集ALL，这将创建一个由Bioconductor定义的特殊类（ExpressionSet）对象。这个类对象含有微阵列数据集的大量信息。有多个相关的方法来处理这种类型的对象。如果需要R给出ALL对象的内容，可以得到以下信息：

```
> ALL
ExpressionSet (storageMode: lockedEnvironment)
assayData: 12625 features, 128 samples
  element names: exprs
phenoData
  sampleNames: 01005, 01010, ..., LAL4  (128 total)
  varLabels and varMetadata description:
    cod:  Patient ID
```

```
    diagnosis:  Date of diagnosis
    ...: ...
    date last seen:  date patient was last seen
    (21 total)
featureData
  featureNames: 1000_at, 1001_at, ..., AFFX-YEL024w/RIP1_at  (12625 total)
  fvarLabels and fvarMetadata description: none
experimentData: use 'experimentData(object)'
  pubMedIds: 14684422 16243790
Annotation: hgu95av2
```

上面输出的 ALL 对象信息分为几个组。首先是含有基因表达水平矩阵的测量数据，该数据包含 128 个样本 12 625 个基因。ALL 对象也含有大量的实验样本的元数据。这包括 phenoData 部分的样本名称和几个相关的协变量的信息。它也包括特征（即基因）信息以及生物医学数据库中有关基因的注释。最后，该对象也包含实验的描述信息。

有多种方法可以用来方便地访问对象 ExpressionSet 中的信息。下面给出几个例子。我们从获取与每一个样本相关联的协变量的信息入手，代码如下：

```
> pD <- phenoData(ALL)
> varMetadata(pD)

                                                                    labelDescription
cod                                                                       Patient ID
diagnosis                                                          Date of diagnosis
sex                                                            Gender of the patient
age                                                        Age of the patient at entry
BT                                            does the patient have B-cell or T-cell ALL
remission        Complete remission(CR), refractory(REF) or NA. Derived from CR
CR                                                           Original remisson data
date.cr                                           Date complete remission if achieved
t(4;11)          did the patient have t(4;11) translocation. Derived from citog
t(9;22)          did the patient have t(9;22) translocation. Derived from citog
cyto.normal                            Was cytogenetic test normal? Derived from citog
citog          original citogenetics data, deletions or t(4;11), t(9;22) status
mol.biol                                                           molecular biology
fusion protein                        which of p190, p210 or p190/210 for bcr/able
mdr                                                             multi-drug resistant
kinet                                                   ploidy: either diploid or hyperd.
ccr                                    Continuous complete remission? Derived from f.u
relapse                                                    Relapse? Derived from f.u
transplant   did the patient receive a bone marrow transplant? Derived from f.u
f.u                                                           follow up data available
date last seen                                                date patient was last seen

> table(ALL$BT)

 B B1 B2 B3 B4  T T1 T2 T3 T4
 5 19 36 23 12  5  1 15 10  2

> table(ALL$mol.biol)

ALL1/AF4  BCR/ABL E2A/PBX1      NEG   NUP-98  p15/p16
      10       37        5       74        1        1

> table(ALL$BT, ALL$mol.bio)

     ALL1/AF4 BCR/ABL E2A/PBX1 NEG NUP-98 p15/p16
  B         0       2        1   2      0       0
  B1       10       1        0   8      0       0
  B2        0      19        0  16      0       1
```

```
B3      0    8    1  14    0    0
B4      0    7    3   2    0    0
T       0    0    0   5    0    0
T1      0    0    0   1    0    0
T2      0    0    0  15    0    0
T3      0    0    0   9    1    0
T4      0    0    0   2    0    0
```

前两个指令用来获取已有的协变量的名称和描述。接着，我们得到样本在两个主要协变量上分布的信息。这两个协变量是：变量 BT，它决定急性淋巴细胞性白血病的类型；变量 mol. bio，描述在每个样本上发现的细胞遗传学异常（NEG 代表没有任何异常）。

我们也可以得到一些关于基因和样本的信息：

```
> featureNames(ALL)[1:10]

 [1] "1000_at"   "1001_at"   "1002_f_at" "1003_s_at" "1004_at"
 [6] "1005_at"   "1006_at"   "1007_s_at" "1008_f_at" "1009_at"

> sampleNames(ALL)[1:5]

[1] "01005" "01010" "03002" "04006" "04007"
```

这段代码给出了开始 10 个基因的名称以及开始 5 个样本的名称。

前面提到过，我们将专注于分析 B 细胞 ALL 个案数据，特别是有基因突变的那个样本子集，基因突变是我们分析的目标类。我们将用下面的代码获得我们将要应用的数据子集：

235 ~ 237

```
> tgt.cases <- which(ALL$BT %in% levels(ALL$BT)[1:5] &
+                    ALL$mol.bio %in% levels(ALL$mol.bio)[1:4])
> ALLb <- ALL[,tgt.cases]
> ALLb

ExpressionSet (storageMode: lockedEnvironment)
assayData: 12625 features, 94 samples
  element names: exprs
phenoData
  sampleNames: 01005, 01010, ..., LAL5  (94 total)
  varLabels and varMetadata description:
    cod:  Patient ID
    diagnosis:  Date of diagnosis
    ...: ...
    date last seen:  date patient was last seen
    (21 total)
featureData
  featureNames: 1000_at, 1001_at, ..., AFFX-YEL024w/RIP1_at  (12625 total)
  fvarLabels and fvarMetadata description: none
experimentData: use 'experimentData(object)'
  pubMedIds: 14684422 16243790
Annotation: hgu95av2
```

第一条指令用于获得我们将考虑的个案的集合。这些个案是变量 BT 和变量 mol. bio 取特定值的样本。前面介绍过函数 table( )，这里调用该函数来显示选择了哪些样本。我们取原始 ALL 对象的子集，得到 94 个样本用于我们的案例研究。这个样本子集只包含变量 BT 和变量 mol. bio 的一部分取值。因此，我们需要更新这两个因子变量的可用水平，并存储为一个新的 ALLb 对象，代码如下：

```
> ALLb$BT <- factor(ALLb$BT)
> ALLb$mol.bio <- factor(ALLb$mol.bio)
```

```
> sapply(levels(ALLb$mol.bio), function(x) summary(as.vector(es[,
+     which(ALLb$mol.bio == x)])))

        ALL1/AF4 BCR/ABL E2A/PBX1    NEG
Min.       2.266   2.195    2.268  1.985
1st Qu.    4.141   4.124    4.152  4.111
Median     5.454   5.468    5.497  5.470
Mean       5.621   5.627    5.630  5.622
3rd Qu.    6.805   6.833    6.819  6.832
Max.      14.030  14.040   13.810 13.950
```

正如我们所看到的，这些样本的子集很相似，并且它们表达水平的全局分布也相似。



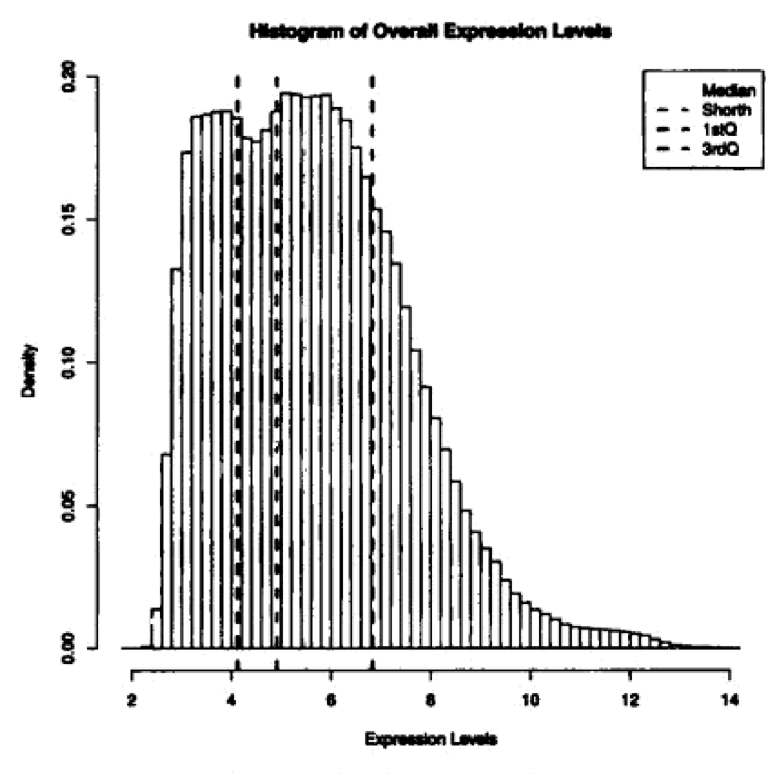


图5-1　基因表达水平的分布

5.3 基因（特征）选择

在许多数据挖掘问题中，特征选择是一项重要的任务。一般的问题是选择数据挖掘问题的特征（或者变量）子集，这个子集与所分析的问题有更高的相关性。可以把特征选择认为是之后建模阶段决定变量的权重（即重要性）这一通用问题的一个实例。一般而言，有两种类型的特征选择方法：1）过滤方法；2）封装方法。在3.3.2节提到，过滤方法应用变量的统计特征来选择最终的特征集合；而封装方法则一般在变量选择过程中要包括数据挖掘模型。过滤方法在一个单一步骤中完成，而封装方法则需要一个搜索过程，我们用迭代方式找到最适合应用的数据挖掘模型的变量子集。特征封装方法明显地需要更多的计算资源，它需要多次运行整个“过滤+模型+评估”周期，直到满足某些收敛准则。这意味着，对于大型的数据挖掘问题，如果时间要求很高，那么封装方法将不适用。然而，也许会找到理论上可行的封装方法，它适用于所应用模型的变量集合。在本节中，我们将描述和应用过滤方法。

5.3.1 基于分布特征的简单过滤方法

我们给出的第一个基因过滤方法是基于基因表达水平分布所给出的信息。这种类型的实验数据通常包含有多个根本不被表达的基因或者变动性极小的基因。第二个特征意味着这些基因不能用来对样本进行区分。而且这种类型的微阵列通常有多个对照探针，它们可以很容易地被剔除。在我们的案例中，应用的微阵列为Affymetric U95Av2，这些探针以字母“AFFX”开头。

ALLb 对象将是本章所使用的数据集。最好把这个对象保存为计算机中的一个本地文件，这样当你需要从头开始分析时，你就不需要重复这些预处理步骤：

```
> save(ALLb, file = "myALL.Rdata")
```

探索数据库

函数 exprs( ) 可以访问基因表达水平矩阵：

```
> es <- exprs(ALLb)
> dim(es)

[1] 12625    94
```

238 该数据集矩阵有 12 625 行（基因/特征），94 列（样本/个案）。

就维数而言，这里最重要的挑战是，对于可以获得的个案（94）有太多的变量（12 625）。对于这种大维度数据，大多数的建模技术将很难获得有意义的结果。在这种情形下，我们的第一个目标是降低变量的数量，即从我们的分析中剔除掉某些基因。为此，下面从探索表达水平数据入手。

下面指令的结果告诉我们，大多数基因表达水平的值是在 4 ~ 7 之间：

```
> summary(as.vector(es))

  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 1.985   4.122   5.469   5.624   6.829  14.040
```

图形方式可以更好地描述表达水平的分布。我们用添加包 genefilter（Gentleman et al.，2010）的一个函数来绘制数据。在应用该添加包之前，必须先安装它。注意，这是一个 bioconductor 添加包，它们不是从标准 R 添加包存储库中安装。最简单的安装生物信息学添加包的方法是通过该项目提供的专用于安装添加包的脚本来完成：

```
> source("http://bioconductor.org/biocLite.R")
> biocLite("genefilter")
```

第一条指令加载安装脚本，第二个指令用它来下载和安装添加包。现在可以进行上述绘图工作，用图形化方式来显示表达水平的分布。

```
> library(genefilter)
> hist(as.vector(es),breaks=80,prob=T,
+      xlab='Expression Levels',
+      main='Histogram of Overall Expression Levels')
> abline(v=c(median(as.vector(es)),
+            shorth(as.vector(es)),
+            quantile(as.vector(es),c(0.25,0.75))),
+        lty=2,col=c(2,3,4,4))
> legend('topright',c('Median','Shorth','1stQ','3rdQ'),
+        lty=2,col=c(2,3,4,4))
```

结果如图 5-1 所示。考虑到我们的数据中有大量的基因表达水平，因此这里改变了直方图函数 hist( ) 的默认区间数，设置参数 breaks 的值为 80，这样就有可能得到分布很详细的逼近。在 239 直方图上，我们绘制了几条竖线来显示中位数、第一个四分位数、第三个四分位数和 SHORTH。统计量 SHORTH 是连续型分布的中心趋势的稳健估计，由添加包 genefilter 中的函数 shorth( ) 实现。该统计量是包含 50% 观测值的中心区间（即四分位距）观测值的均值。

基因变异样本的基因表达水平分布是否和其他样本的分布不同呢？下面的代码回答了这个问题。

我们可以通过下面的图形来得到每个基因在所有样本上的总体分布情况。我们用中位数和四分位距（IQR）来代表基因的这些分布。下面的代码用来得到每个基因的这些统计量值并绘制它们的散点图，如图 5-2 所示。

```
> rowIQRs <- function(em)
+   rowQ(em,ceiling(0.75*ncol(em))) - rowQ(em,floor(0.25*ncol(em)))
> plot(rowMedians(es),rowIQRs(es),
+       xlab='Median expression level',
+       ylab='IQR expression level',
+       main='Main Characteristics of Genes Expression Levels')
```

241 R 添加包 Biobase 中的函数 rowMedians( ) 用来得到矩阵每行向量的中位数。这种方式获取中位数是很有效的。另一种方法是应用函数 apply( )，它没有这里的方法有效[⊖]。函数 rowQ( ) 是添加包 Biobase 提供的另一个高效率的函数，它用来得到矩阵每行向量分布的四分位数，该函数的第二个参数是一个取值在 1 到矩阵列数之间的整数（1 代表最小值，其他值代表给出最大值）。这里，我们应用函数 rowQ( ) 的第 3 个四分位数减去第 1 个四分位数，从而得到四分位距（IQR）。这些统计量分别对应于 75% ~25% 的数据[⊜]。在上面的代码中，我们应用函数 floor( ) 和函数 ceiling( ) 得到矩阵每一行数值的相应四分位数的位置。这两个函数采用不同的舍入方法来获取一个浮点数的整数部分。读者可以试验这两个函数，理解它们的不同之处。我们应用函数 rowQ( ) 创建了函数 rowIQRs( )，从而可以获取每行的四分位距（IQR）。

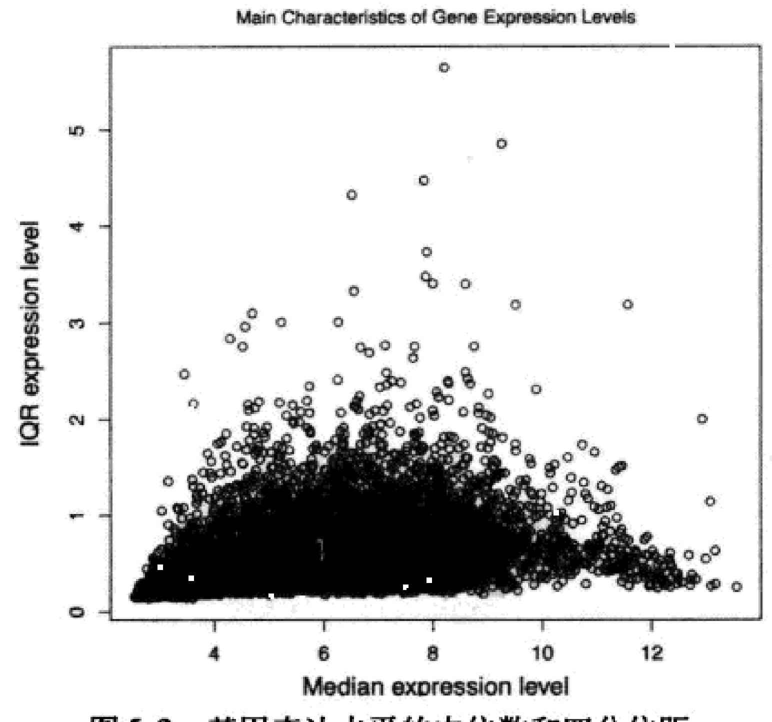


图 5-2 基因表达水平的中位数和四分位距

图 5-2 给出了有趣的信息。从图 5-2 中可知，很大比例的基因的变动性很小（IQR 接近 0）。前面提到，如果一个基因在所有样本上的变动性很小，我们就有很好地理由认为它不能很好地起到区分不同类型的 B 细胞 ALL 变异的作用。这就意味着我们可以安全地把这些基因从我们的 242 分类任务中剔除。我们需要注意的是，这里的推理有一个缺陷。事实上，这里是从单个基因来观察的。因此上述推理的风险在于，如果我们把某些在所有样本上有很小变化的基因和其他基因放到一起，那么它们有可能对分类任务起作用。然而，对于这样高维度的数据集，探索基因之间的相互作用是很困难的。因此，我们这里单独考虑每个基因的方法仍然是处理这类问题最常见

⊖ 作为一个练习，读者可以对这两个函数应用 system. time( ) 函数，观察它们在运行效率上的区别。

⊜ 在所有的数据中，75% 的数据点小于第 3 个四分位数相应的点，25% 的数据点小于第 1 个四分位数相应的点。——译者注

的方法。不过，也有其他方法在考虑基因之间相互依赖的基础上来估计特征的重要性。例如，RELIEF 方法（Kira and Rendel, 1992；Kononenko et al. , 1997）。

我们这里用尝试的四分位距（IQR）界限值来剔除那些变动性很小的基因。即，我们将剔除变动性小于总体 IQR 1/5 的所有基因。添加包 genefilter 中的函数 nsFilter( ) 可以用于这种过滤任务。代码如下：

```
> library(genefilter)
> ALLb <- nsFilter(ALLb,
+                  var.func=IQR,
+                  var.cutoff=IQR(as.vector(es))/5,
+                  feature.exclude="^AFFX")
> ALLb

$eset
ExpressionSet (storageMode: lockedEnvironment)
assayData: 4035 features, 94 samples
  element names: exprs
phenoData
  sampleNames: 01005, 01010, ..., LAL5  (94 total)
  varLabels and varMetadata description:
    cod:  Patient ID
    diagnosis:  Date of diagnosis
    ...: ...
    mol.bio:  molecular biology
    (22 total)
featureData
  featureNames: 41654_at, 35430_at, ..., 34371_at  (4035 total)
  fvarLabels and fvarMetadata description: none
experimentData: use 'experimentData(object)'
  pubMedIds: 14684422 16243790
Annotation: hgu95av2

$filter.log
$filter.log$numLowVar
[1] 4764

$filter.log$numDupsRemoved
[1] 2918

$filter.log$feature.exclude
[1] 19

$filter.log$numRemoved.ENTREZID
[1] 889
```

从结果中可知，过滤后的基因从原来的 12 625 个变为剩下的 4035 个。这是相当显著的减少。考虑到我们只有 94 个观测值，因此对于我们的分类任务来说，过滤后的基因个数仍然太大。

函数 nsFilter( ) 的结果是一个含有多个元素的列表。其中的多个元素包含被剔除基因的信息，而且元素 eset 含有选出的基因对象。有了过滤的结果之后，我们可以用选出的对象来更新 ALLb 对象和 es 对象，代码如下：

```
> ALLb <- ALLb$eset
> es <- exprs(ALLb)
> dim(es)

[1] 4035   94
```

5.3.2 ANOVA 过滤

如果一个基因表达水平的分布在目标变量的所有可能值上类似，则可以确定这个基因无助于区分这些目标变量值。我们的下一个方法就是基于这点。我们将比较同一类 B 细胞 ALL 突变样本所在数据子集的基因表达水平的均值，即在同一目标变量值条件下的基因均值。如果统计上有很大信心认为一个基因的表达水平均值在属于每一类突变的样本组上没有显著区别，那么这些基因可以被排除在进一步分析之外。

可以用因子分析（ANOVA）来比较多于两个的组的均值。在我们的案例研究中，我们有 4 组案例，每一组相应于一个类型的 B 细胞 ALL 基因突变。借助于 R 的添加包 genefilter，可以在 R 中很容易地进行这种基于 ANOVA 的过滤。这种类型过滤的代码如下：

```
> f <- Anova(ALLb$mol.bio, p = 0.01)
> ff <- filterfun(f)
> selGenes <- genefilter(exprs(ALLb), ff)

> sum(selGenes)

[1] 752

> ALLb <- ALLb[selGenes, ]
> ALLb
ExpressionSet (storageMode: lockedEnvironment)
assayData: 752 features, 94 samples
  element names: exprs
phenoData
  sampleNames: 01005, 01010, ..., LAL5  (94 total)
  varLabels and varMetadata description:
    cod:  Patient ID
    diagnosis:  Date of diagnosis
    ...: ...
    mol.bio:  molecular biology
    (22 total)
featureData
  featureNames: 266_s_at, 33047_at, ..., 40698_at  (752 total)
  fvarLabels and fvarMetadata description: none
experimentData: use 'experimentData(object)'
  pubMedIds: 14684422 16243790
Annotation: hgu95av2
```

函数 Anova( ) 创建了一个新的函数来执行 ANOVA（方差分析）过滤，它需要一个因子来给出数据集的组别和一个统计显著性水平。该函数的结果保存在变量 f 中。函数 filterfun( ) 的原理类似，它给出一个用于表达矩阵的过滤函数。它应用函数 genefilter( ) 产生一个向量，向量的元素为逻辑值，向量中元素的个数和给定的表达矩阵中基因的个数相同。按照 ANOVA 统计检验，如果向量的值为 TRUE，则认为相应的基因是有用的。从结果可知，只有 752 个有用基因。最后，可以用这个向量来对 ExpressionSet 对象进行过滤。图 5-3 给出了 ANOVA 检验所选出的基因的中位数和四分位距。下面的代码用于绘制图 5-3：

```
> es <- exprs(ALLb)
> plot(rowMedians(es),rowIQRs(es),
+      xlab='Median expression level',
+      ylab='IQR expression level',
+      main='Distribution Properties of the Selected Genes')
```

从图 5-3 可知，用中位数和四分位距来衡量的基因的变化性提供了证据，表明这些基因表达

的取值的标度很不相同。如果描述记录的一组变量的标度不同，许多建模技巧将会导致问题。也就是说，那些依赖于记录之间距离的建模方法将会有问题，因为距离函数一般是变量之间差值的总和。因此，如果变量的标度相差很大，那么具有较大均值的变量将对样本间的距离有较大的影响。为了避免这个问题，一般要对数据进行标准化，即减去变量的典型取值，然后除以变量的一个变动性指标。考虑到不是所有的建模技术都会受到变量不同标度的影响，因此我们把变量的标准化问题放到建模阶段，这样是否进行标准化将依赖于所应用的工具。

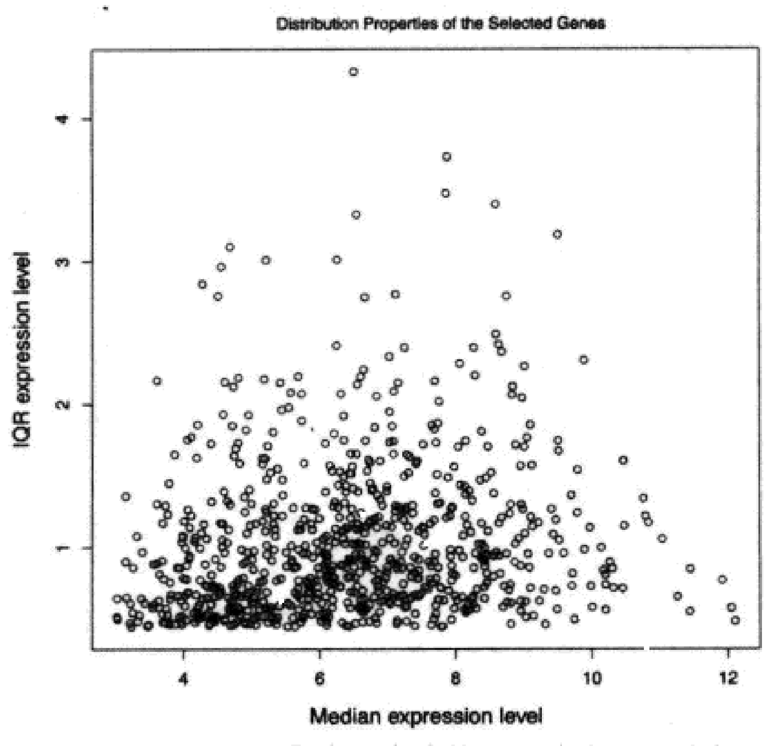


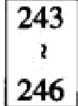
图5-3 最终的基因集合的中位数和四分位距的散点图

5.3.3 用随机森林进行过滤

尽管从 ANOVA 过滤得到的表达水平矩阵的变量个数还是多于观测值的个数，但这已经在可以建模的范围之内。实际上，我们在5.4节就应用这个矩阵进行建模。然而，有人可能要问，是否有更好的方法，从而得到具有更“标准”维数的数据集。事实上，我们可以尝试进一步减少特征的个数，然后比较用不同的数据集得到的结果。

可以用随机森林得到变量对分类任务有用程度的排序。在3.3.2节的预测问题中，我们用随机森林得到了变量对预测任务重要性的排序。

在演示随机森林方法之前，先更改基因的名称。现在基因的名称不是许多建模技术所应用的数据框所期望的标准名称。可以应用函数 make.names() 来“解决”这个问题，代码如下：


243 ~ 246

```
> featureNames(ALLb) <- make.names(featureNames(ALLb))
> es <- exprs(ALLb)
```

函数 featureNames() 用来获取 ExpressionSet 对象中基因的名称。

应用下面的代码，使用随机森林来获取基因的排序：

```
> library(randomForest)
> dt <- data.frame(t(es), Mut = ALLb$mol.bio)
> rf <- randomForest(Mut ~ ., dt, importance = T)
> imp <- importance(rf)
> imp <- imp[, ncol(imp) - 1]
> rf.genes <- names(imp)[order(imp, decreasing = T)[1:30]]
```

我们把基因突变信息加入到表达矩阵的转置矩阵中，从而构造出训练集数据[⊖]。然后调用随

⊖ 注意表达矩阵的行代表基因（即变量）。

机森林函数，把参数 importance 设为 TRUE，这样就得到了变量重要性的估计值。函数 importance( ) 用来得到每个变量和目标变量的相关程度，按照不同的准则，它在不同的列返回每一个分类值的分数。其中一个列用来衡量当每个变量被依次剔除后，分类精确度平均减少的估计值，选择该列作为变量的重要性分值。最后，选择出现在变量重要性分值前 30 的基因。

我们可能想知道这 30 个基因的表达水平在所有不同突变类型上的分布情况。可以用下面的代码获得这 30 个基因的中位数水平：

```
> sapply(rf.genes, function(g) tapply(dt[, g], dt$Mut, median))

         X40202_at X1674_at X1467_at X1635_at X37015_at X34210_at
ALL1/AF4  8.550639 3.745752 3.708985 7.302814  3.752649  5.641130
BCR/ABL   9.767293 5.833510 4.239306 8.693082  4.857105  9.204237
E2A/PBX1  7.414635 3.808258 3.411696 7.562676  6.579530  8.198781
NEG       7.655605 4.244791 3.515020 7.324691  3.765741  8.791774
         X32116_at X34699_at X40504_at X41470_at X41071_at X36873_at
ALL1/AF4  7.115400  4.253504  3.218079  9.616743  7.698420  7.040593
BCR/ABL   7.966959  6.315966  4.924310  5.205797  6.017967  3.490262
E2A/PBX1  7.359097  6.102031  3.455316  3.931191  6.058185  3.634471
NEG       7.636213  6.092511  3.541651  4.157748  6.573731  3.824670
         X35162_s_at X38323_at X1134_at X32378_at X1307_at X1249_at
ALL1/AF4    4.398685  4.195967 7.846189  8.703860 3.368915 3.582763
BCR/ABL     4.924553  4.866452 8.475578  9.694933 4.945270 4.477659
E2A/PBX1    4.380962  4.317550 8.697500 10.066073 4.678577 3.257649
NEG         4.236335  4.286104 8.167493  9.743168 4.863930 3.791764
         X33774_at X40795_at X36275_at X34850_at X33412_at X37579_at
ALL1/AF4  6.970072  3.867134  3.618819  5.426653 10.757286  7.614200
BCR/ABL   8.542248  4.544239  6.259073  6.898979  6.880112  8.231081
E2A/PBX1  7.385129  4.151637  3.635956  5.928574  5.636466  9.494368
NEG       7.348818  3.909532  3.749953  6.327281  5.881145  8.455750
         X37225_at X39837_s_at X37403_at X37967_at X2062_at X35164_at
ALL1/AF4  5.220668    6.633188  5.888290  8.130686 9.409753  5.577268
BCR/ABL   3.460902    7.374046  5.545761  9.274695 7.530185  6.493672
E2A/PBX1  7.445655    6.708400  4.217478  8.260236 7.935259  7.406714
NEG       3.387552    6.878846  4.362275  8.986204 7.086033  7.492440
```

从结果中可以观测到，不同类型突变的基因的中位数表达水平的区别，它也给出了这些基因用于分类的区别能力。我们可以用图形方式来检查这 94 个样本的具体基因表达水平的值，可以获取更多的细节：

```
> library(lattice)
> ordMut <- order(dt$Mut)
> levelplot(as.matrix(dt[ordMut,rf.genes]),
+           aspect='fill', xlab='', ylab='',
+           scales=list(
+             x=list(
+               labels=c('+','-','*','|')[as.integer(dt$Mut[ordMut])],
+               cex=0.7,
+               tck=0)
+             ),
+           main=paste(paste(c('"+"','"-"','"*"','"|"'),
+                            levels(dt$Mut)
+                            ),
+                      collapse='; '),
+           col.regions=colorRampPalette(c('white','orange','blue'))
+           )
```

上面代码获得的图形如图 5-4 所示。注意，有几个基因在不同突变类型上的表达水平显著不同。例如，基因 X36275_at 的表达水平在突变 ALL1/AF4 和突变 BCR/ABL 之间显著不同。在上面的绘图代码中，我们应用了添加包 lattice 的函数 levelplot( )。该函数可以绘制一个数值矩阵的彩色图形。这里我们用这个函数来绘制按照突变类型来排序的样本表达矩阵。

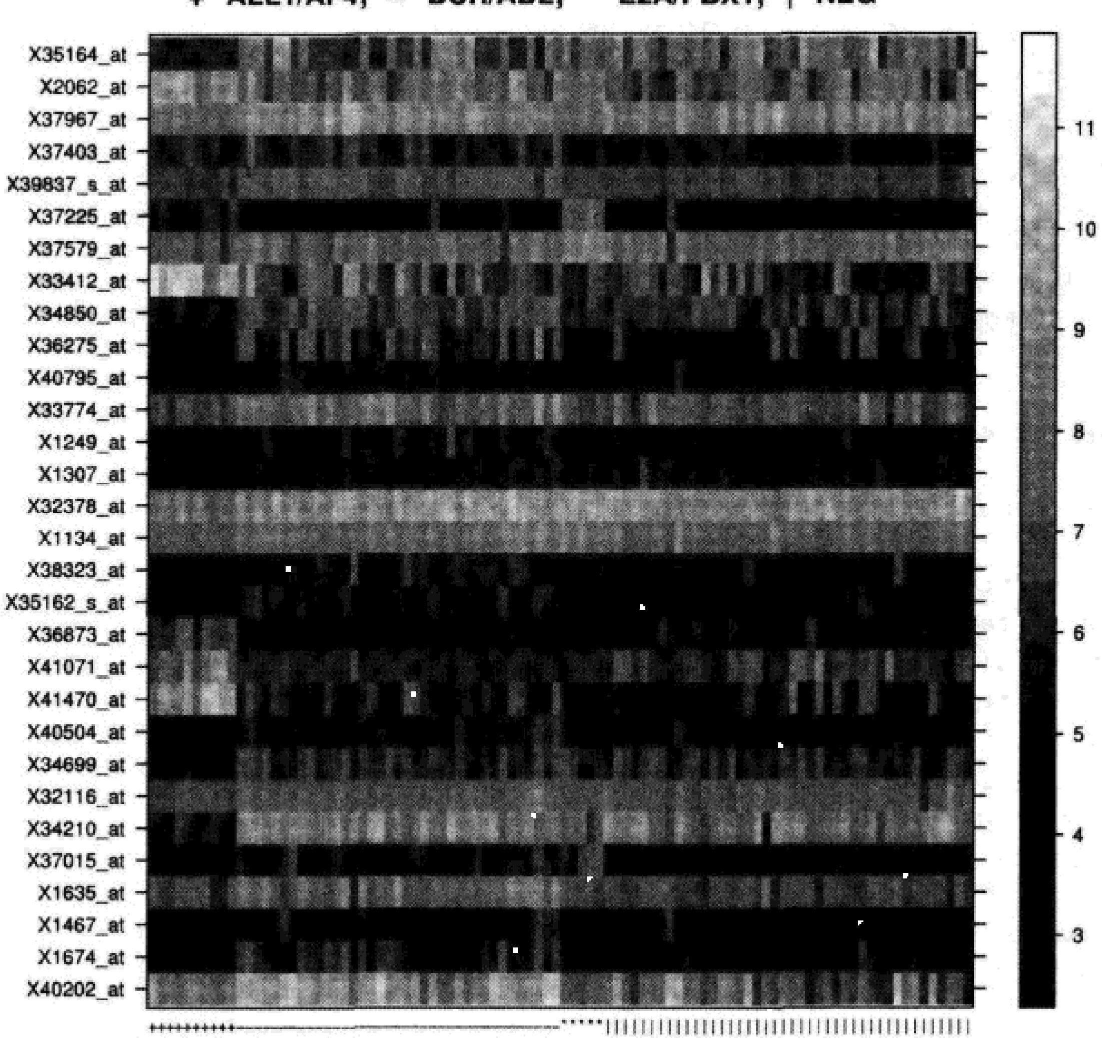


图 5-4　94 个样本的 30 个基因的表达水平

5.3.4　用特征聚类的组合进行过滤

本节用聚类算法得到了 30 个变量类，假设每个类中的变量相似。用这 30 个变量类来得到组合分类模型。组合分类模型由 m 个模型构成，每个模型是由 30 个变量来获得的，其中每个变量是从这 30 个变量类中随机选择的。

组合方法是一种学习方法，它构建一组组合模型，然后用这组模型对新记录的预测值的某种形式的平均来对该记录进行分类。就目前所知，组合方法的性能常常超过组成它的单个模型。组合模型基于单个模型间的某种形式的多样性。有多种形式来创建这种多样性的模型。例如，可以通过不同的模型参数，或者每个模型用不同的训练集数据来获得不同的模型。另一种多样性方式是用不同的预测变量来得到组合中的每个模型。本节的组合方法将采用后一种方法。如果用于获取聚类的原始预测变量集合有高度的冗余，那么这种方法将很奏效。假设 ANOVA 过滤得到的变量有某种程度的冗余。我们通过变量聚类对这种冗余性进行建模。聚类方法基于距离，本案例中是变量间的距离。如果两个变量的表达水平在 94 个样本上相似，那么这两个变量就是接近的（所以它们是相似的）。通过变量聚类，我们期望找到相互类似的基因组。添加包 Hmisc 有一个函数实现了数据集变量的层次聚类法，该函数是 varclus( )。下面给出用该函数进行变量聚类的代码：

```
> library(Hmisc)
> vc <- varclus(t(es))
> clus30 <- cutree(vc$hclust, 30)
> table(clus30)

clus30
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
27 26 18 30 22 18 24 46 22 20 24 18 56 28 47 32 22 31 18 22 18 33 20 20 21
26 27 28 29 30
17  9 19 30 14
```

我们应用函数 cutree( ) 得到了由 30 个变量组形成的聚类。然后，检查每个组中有多少变量（即基因）。基于这个聚类，我们可以通过在每个变量组中随机选取一个变量来构成我们的预测变量集合。理由是每一个变量组的成员变量是类似的，因此有某种程度的冗余。

为了方便通过随机抽样从选定个数的变量组（默认 30 组）中得到一组变量，下面的函数实现了这个过程：

```
> getVarsSet <- function(cluster,nvars=30,seed=NULL,verb=F)
+ {
+   if (!is.null(seed)) set.seed(seed)
+
+   cls <- cutree(cluster,nvars)
+   tots <- table(cls)
+   vars <- c()
+   vars <- sapply(1:nvars,function(clID)
+     {
+       if (!length(tots[clID])) stop('Empty cluster! (',clID,')')
+       x <- sample(1:tots[clID],1)
+       names(cls[cls==clID])[x]
+     })
+   if (verb)  structure(vars,clusMemb=cls,clusTots=tots)
+   else       vars
+ }
> getVarsSet(vc$hclust)

 [1] "X41346_at"   "X33047_at"   "X1044_s_at"  "X38736_at"   "X39814_s_at"
 [6] "X649_s_at"   "X41672_at"   "X36845_at"   "X40771_at"   "X38370_at"
[11] "X36083_at"   "X34964_at"   "X35228_at"   "X40855_at"   "X41038_at"
[16] "X40495_at"   "X40419_at"   "X1173_g_at"  "X40088_at"   "X879_at"
[21] "X39135_at"   "X34798_at"   "X39649_at"   "X39774_at"   "X39581_at"
[26] "X37024_at"   "X32585_at"   "X41184_s_at" "X33305_at"   "X41266_at"

> getVarsSet(vc$hclust)

 [1] "X40589_at"   "X33598_r_at" "X41015_at"   "X38999_s_at" "X37027_at"
 [6] "X32842_at"   "X37951_at"   "X35693_at"   "X36874_at"   "X41796_at"
[11] "X1462_s_at"  "X31751_f_at" "X34176_at"   "X40855_at"   "X1583_at"
[16] "X38488_s_at" "X32542_at"   "X32961_at"   "X32321_at"   "X879_at"
[21] "X38631_at"   "X37718_at"   "X948_s_at"   "X38223_at"   "X34256_at"
[26] "X1788_s_at"  "X38271_at"   "X37610_at"   "X33936_at"   "X36899_at"
```

每次调用这个函数，都得到新的一组 30 个变量。通过这个函数，可以容易得到由不同的预测变量所构成的一组数据集，然后用每一个数据集进行建模。在 5.4 节，我们给出用这种策略来获取组合模型的函数。

特征选择算法的参考文献

在许多学科中，特征选择都是一个充分研究的主题。在数据挖掘领域中有关该主题的优秀

概述文章和参考资料是 Liu 和 Motoda（1998）、Chizi 和 Maimon（2005）以及 Wettschereck 等（1997）。

5.4 遗传学异常的预测

本节描述用于预测任务的整个建模过程，预测 B 细胞 ALL 样本的遗传学异常类型。

5.4.1 定义预测任务

我们面对的数据挖掘问题是一个预测问题。精确地说，它是一个分类问题。预测分类任务包含用一组预测变量的信息来预测一个名义目标变量取值的建模过程。用一组已经知道其所研究的对象所属分类的观测值来得到模型，也就是说，这些观测值的预测变量的值和目标变量的值是已知的。

247～251

在本案例中，目标变量是 B 细胞 ALL 样本的遗传学异常分类。在我们所选的数据集中，这个变量有 4 个可能的取值：ALL1/AF4、BCR/ABL、E2A/PBX1 和 NEG。预测变量由一组选定的基因构成，基因的表达水平是已知的。在建模过程中，基于 5.3 节的内容，我们用选定基因的不同集合来尝试建模。这意味着随着这些实验的不同，预测变量的个数（特征）会变化。观测值由 94 个 B 细胞 ALL 个案构成。

5.4.2 模型评价标准

预测任务是一个分类问题。预测分类模型通常用误分率或考它的对立面——正确率来衡量。然而，也有其他的评价预测分类模型的标准，例如在 ROC 曲线下的面积、配对的指标（即决策精确度和回溯精确度）以及类概率估计的正确率（即，Brier 分数）。R 的添加包 ROCR 给出了这些评价标准很好的例子。

一个问题评价标准的选择常常取决于用户的目的。由于信息不完整，选择评价标准常常是一个困难的决策，例如缺乏错误地把本来属于类 i 的个案分类到类 j 损失的信息（误分类信息）。

在本章的案例研究中，我们没有误分类损失的信息。因此，这里假设每个误分类的严重程度是相同的，例如把一个属于类 E2A/PBX1 突变误分为类 NEG 的损失和把类 ALL1/AF4 误分为类 BCR/ABL 的损失相等。另外，我们有两个以上的分类，而 ROC 分析还没有很好地推广到多类问题。另外，近来发现应用 ROC 曲线下的面积有缺陷问题（Hand，2009）。基于上面的分析，我们用标准的准确率评价标准，定义为：

$$\overline{\text{acc}} = 1 - \frac{1}{N}\sum_{i=1}^{N} L_{0/1}(y_i, \hat{y}_i) \tag{5-1}$$

这里 N 是测试样本的大小，而 $L_{0/1}()$ 是损失函数，其定义如下：

252

$$L_{0/1}(y_i, \hat{y}_i) = \begin{cases} 0 & y_i = \hat{y}_i \\ 1 & \end{cases} \tag{5-2}$$

5.4.3 实验过程

这里的数据集中观测值的个数很少：只有 94 个个案。这种情况下，得到这类问题误分率可靠估计的较好实验方法是弃一交叉验证法（LOOCV）。LOOCV 是我们前面应用的 k 折交叉验证实验方法的一个特例，即 k 为观测值的个数。简单地说，LOOCV 从 $N-1$ 个个案获取一个模型，一共获得 N 个模型（N 为数据集的大小），每个模型由那个没有用于建模的观测值来进行验证。本书提供的添加包函数 loocv( ) 实现了这种实验。这个函数应用的过程与前面章节描述的其他实验比较中的函数类似。下面的代码用数据集 iris 来演示该函数的应用：

```
> data(iris)
> rpart.loocv <- function(form,train,test,...) {
```

```
+   require(rpart,quietly=T)
+   m <- rpart(form,train,...)
+   p <- predict(m,test,type='class')
+   c(accuracy=ifelse(p == resp(form,test),100,0))
+ }
> exp <- loocv(learner('rpart.loocv',list()),
+              dataset(Species~.,iris),
+              loocvSettings(seed=1234,verbose=F))

> summary(exp)

== Summary of a Leave One Out Cross Validation  Experiment ==

 LOOCV experiment with verbose =  FALSE  and seed = 1234

* Dataset ::  iris
* Learner  ::  rpart.loocv  with parameters:
* Summary of Experiment Results:

         accuracy
avg      93.33333
std      25.02795
min       0.00000
max     100.00000
invalid   0.00000
```

函数 loocv( ) 采用 3 个常见的参数：模型、数据集和实验的设置。它返回一个 loocvRun 类对象，我们可以用函数 summary( ) 来获取该类对象的结果。

253

用户定义的函数（上例中为 rpart. loocv( )）运行模型，应用模型获取测试集的预测值，返回希望 LOOCV 所估计的评估指标向量。在上面的例子中，它简单地计算模型的正确率。这里要记住，在 LOOCV 过程中，每一个实验迭代过程的测试集是由一个单一的观测值构成，所以这里我们只要检查预测值是否等于真实值。

5.4.4 建模技术

如前面所描述，我们将应用 3 个不同的数据集，它们所应用的预测变量是不同的。一个数据集应用 ANOVA 过程选择的所有基因，另外两个从这些基因中选择 30 个。所有的数据集都包含 94 个 B 细胞 ALL 个案。除目标变量外，所有信息都是数值型的。

为了解决这类问题，我们应用三种不同的建模技术。在本书的前面章节中，我们已经应用过其中的两种，它们是随机森林和支持向量机（SVM）。它们被认为是最好用的预测方法之一。本节将尝试的第三个方法是一个新的算法，它是一种基于观测值之间距离的方法，即 k 近邻方法。

基于随机森林的模型特别适合于处理有大量特征的问题。这个特征来源于随机森林方法所应用的算法，它从问题的所有原始变量中随机选择一个子集进行建模（参见 5.4.4.1 节）。而选择 k 近邻方法，其动机是基于这样的假定：同一类型的突变样本有类似的基因“签名”，即在用于描述这些样本的基因上有类似的表达值。这个假定的有效性极大地依赖于所选定的描述样本的基因。也就是说，这些基因应该能够很好地区分不同的突变类型。在 5.4.4.2 节可以看到，k 近邻方法利用了个案之间的相似性，因此应该满足上面的假定。最后，支持向量机用来尝试找到基因表达和遗传学异常可能存在的非线性关系。

3.4.2.2 节描述了支持向量机，它们是高度非线性的模型，可以用于回归和分类问题。而且，在 R 的多个不同的支持向量机实现中，我们选用添加包 e1071 的 svm( ) 函数。

5.4.4.1 随机森林

随机森林是组合模型的一个例子（Breiman，2001），也就是说，它是由一组较简单模型形成的。特别地，随机森林由一组决策树构成，取决于分析的问题采用回归树还是分类树。用户决定组合中树的数量。每棵树都是通过自助法抽样进行训练的，自助法抽样从原始数据集中用有放回抽样法随机抽取 N 个个案，这里 N 是数据集中个案的个数。每一个这样的训练集获得一个不同的树。仅仅考虑原始问题预测变量的一个随机子集来决定这些树的每一个结点，这些随机子集大小应该大大小于数据集预测变量的个数。这些树完全生长，也就是说，它们没有任何事后剪枝。有关如何获得基于树的模型的细节，参见2.6.2节。 254

采用每棵树的预测值的平均值作为这些组合的预测值。对于分类问题，则采用投票机制。对于回归问题，对每一棵树的预测值进行平均得到随机森林的预测值。

在R中，随机森林由添加包 randomForest 实现。本书已经给出了多个应用该包中的随机森林函数来进行特征选择的例子。

随机森林模型的参考文献

随机森林模型可以参考 Breiman（2001）的原始文献。也可以访问网站 http://stat-www.berkeley.edu/users/breiman/RandomForests/得到随机森林模型的更多资料。

5.4.4.2 *k* 近邻方法

k 近邻方法属于懒惰学习算法。这类算法实际上没有从训练数据得到一个模型。它们简单地存储这个训练集数据。算法的主要工作在于预测时。给一个新的测试个案，它会在存储的训练集中寻找类似的个案作为预测值。k 个最相似的训练集个案被用来得到给定测试个案的预测值。在分类问题中，预测值通过投票方法得到，因此选用的 k 值最好为奇数。然而，一些更精细的投票方法也可能会考虑到测试个案与 k 个近邻中每一个的距离。对于回归问题，它采用这 k 个邻近个案目标变量的均值而不是投票。

这类模型严重依赖于个案之间相似性的标记方式。这类相似性的标记通常由预测变量所定义的输入空间上的一个测度来定义。这个测度是一个距离函数，它用来计算代表任何两个观测值之间差异的一个数值。有多个距离函数，常见的是欧氏距离函数，定义如下： 255

$$d(\boldsymbol{x}_i,\boldsymbol{x}_j) = \sqrt{\sum_{k=1}^{p}(X_{i,k} - X_{j,k})^2} \qquad (5\text{-}3)$$

这里 p 代表预测变量的个数，$\boldsymbol{x}_i$ 和 $\boldsymbol{x}_j$ 是两个观测值。

这里的距离对于所选择的变量标度和变量的相关性很敏感，它们可能扭曲相似性标记。另外，变量的标度应该是一致的，否则可能低估一些较低均值的变量间的差异。

选择近邻个案数 k 也是这类方法的一个重要参数。常见的 k 值选择为｛1，3，5，7，11｝，但这仅仅是经验值。但是，要避免选择较大的 k 值，这样有可能会选用远离测试个案的样本。显然，这取决于训练集数据的密度。太稀疏的数据集更可能导致这种风险。和其他训练模型一样，“理想”的参数选择可以通过一些实验方法来估计。在R中，添加包 class（Venables and Ripley，2002）中所包含的函数 knn( ) 实现了 k 近邻方法。下面用数据集 iris 给出这个函数的一个示例：

```
> library(class)
> data(iris)
> idx <- sample(1:nrow(iris), as.integer(0.7 * nrow(iris)))
> tr <- iris[idx, ]
> ts <- iris[-idx, ]
> preds <- knn(tr[, -5], ts[, -5], tr[, 5], k = 3)
> table(preds, ts[, 5])
```

```
preds        setosa versicolor virginica
  setosa         14          0         0
  versicolor      0         14         2
  virginica       0          1        14
```

从上面代码可知，函数 knn( ) 应用的是一个非标准的用户界面。第一个参数是除了目标变量所在列以外的训练集数据；第二个参数是测试集，同样不包含目标变量值；第三个参数是训练集数据的目标变量值；最后是其他几个控制本方法的参数，其中参数 k 控制近邻的个数。可以创建一个函数以更加标准的公式形式的界面来应用函数 knn( )，代码如下：

```
> kNN <- function(form, train, test, norm = T, norm.stats = NULL,
+     ...) {
+     require(class, quietly = TRUE)
+     tgtCol <- which(colnames(train) == as.character(form[[2]]))
+     if (norm) {
+         if (is.null(norm.stats))
+             tmp <- scale(train[, -tgtCol], center = T, scale = T)
+         else tmp <- scale(train[, -tgtCol], center = norm.stats[[1]],
+             scale = norm.stats[[2]])
+         train[, -tgtCol] <- tmp
+         ms <- attr(tmp, "scaled:center")
+         ss <- attr(tmp, "scaled:scale")
+         test[, -tgtCol] <- scale(test[, -tgtCol], center = ms,
+             scale = ss)
+     }
+     knn(train[, -tgtCol], test[, -tgtCol], train[, tgtCol],
+         ...)
+ }
> preds.norm <- kNN(Species ~ ., tr, ts, k = 3)
> table(preds.norm, ts[, 5])

preds.norm   setosa versicolor virginica
  setosa         14          0         0
  versicolor      0         14         3
  virginica       0          1        13

> preds.notNorm <- kNN(Species ~ ., tr, ts, norm = F, k = 3)
> table(preds.notNorm, ts[, 5])

preds.notNorm setosa versicolor virginica
   setosa         14          0         0
   versicolor      0         14         2
   virginica       0          1        14
```

上述函数允许用户指明是否在调用函数 knn( ) 之前对数据进行标准化。这通过参数 norm 来完成。在上面的例子中，有两个应用它的例子。第三个选择是给参数 norm. stats 提供一个两元素的列表，列表的两个元素分别给出中心化和离散程度统计量。如果没有应用参数 norm. stats，那么函数将应用均值作为中心的估计值，用标准差作为离散程度的估计值。在我们的实验中，我们将用中位数和四分位距作为参数来调用函数。在本书提供的 R 添加包中包含了函数 kNN( )，因此你不必手工输入以上代码。

***k* 近邻方法的参考文献**

这类方法的标准参考文献是 Cover 和 Hart（1967）的文章。该方法很好的概述可以参考 Aha 等（1991）的文章以及 Aha（1997）的文章。更深入的分析可以参考 Aha（1990）的博士论文以及 Wettschereck（1994）的文章。一个不同于懒惰学习但是相关的方法是所谓的“局部模型”

(Nadaraya, 1964; Watson, 1964)。这方面的优秀文献有 Atkeson 等 (1997) 的文章, Cleveland 和 Loader (1996) 的文章。

256 ~ 257

5.4.5 模型比较

本节描述应用 LOOCC 实验方法来比较所选模型的过程。

在5.3节，我们学习了几个特征选择的例子。我们用基本的过滤法删除了较小变化的基因和对照探针。下一步，我们应用基于表达水平在目标变量上的条件分布方法，该方法是基于 ANOVA 统计检验。最后，从 ANOVA 检验的结果中，我们尝试应用随机森林和变量聚类方法来进一步减少基因的数目。除了第一个简单的过滤法外，所有其他方法都某种程度依赖于目标变量的值。我们可能怀疑这些过滤阶段是否可以在实验比较之前进行，或者是否可以把这些过滤过程集成到比较过程中。如果目的是获得对新样本分类正确率的无偏估计，那么我们应该把这些过滤过程作为需要评估和比较的数据挖掘过程的一部分。否则，它意味着我们得到的估计是有偏的，因为应用测试集信息去获得建立模型的基因。事实上，如果我们用整个数据集来决定应用哪些基因，那么在这个选择过程中就应用了那些作为测试集的一部分信息，这些信息应该是未知的。基于此，我们把过滤阶段包含进用户定义的函数中，这些函数实现了我们要比较的模型。

对于 LOOCV 过程的每一次迭代，在创建预测模型之前，只采用 LOOCV 函数提供的训练集进行特征选择过程。初始的过滤过程将在 LOOCV 比较之前进行，如果我们在 LOOCV 过程之内进行简单过滤过程，那么这一步骤中剔除的基因将不发生变化。对照探针总是被剔除，如果没有给出一个单一样本，那么由于低方差被剔除的基因将仍旧非常可能被剔除（这是 LOOCV 过程每一次迭代所做的）。

我们将描述为运行实验需要提供给函数 LOOCV 的用户定义函数。对于每一个建模技术，我们将评估几个变体。这些变体不仅在模型本身的几个参数上有所不同，而且它们所应用的特征选择过程也不相同。下面的列表给出了每个建模技术的这些变体的信息。

```
> vars <- list()
> vars$randomForest <- list(ntree=c(500,750,100),
+                          mtry=c(5,15,30),
+                          fs.meth=list(list('all'),
+                                       list('rf',30),
+                                       list('varclus',30,50)))
> vars$svm <- list(cost=c(1,100,500),
+                 gamma=c(0.01,0.001,0.0001),
+                 fs.meth=list(list('all'),
+                              list('rf',30),
+                              list('varclus',30,50)))
> vars$knn <- list(k=c(3,5,7,11),
+                 norm=c(T,F),
+                 fs.meth=list(list('all'),
+                              list('rf',30),
+                              list('varclus',30,50)))
```

上面的列表含有3个元素，每个元素对应于一个需要比较的算法。对于每个模型，该列表包含了应该用到的参数。对于每一个参数，给出了一组可选值。所有这些值的组合将决定系统的不同变体。对于随机森林，考虑参数 ntree 的3个值，它设定组合中树的个数。参数 mtry 有3个值，当用于确定每个树结点的检验时，该参数决定变量的随机子集的大小。最后一个参数 fs. meth 提供了我们下面将描述的特征选择阶段的选项。对于支持向量机，参数 cost 和参数 gamma 都考虑3个取值。对于 k 近邻方法，考虑参数 k 的4个值；对于决定是否标准化预测变量数据的参数，考

些变体的实验需要很长一段时间[⊖]。除非你意识到时间的限制，否则这里不建议你运行下面的实验。实验的结果在本书的网站上，你可以不用运行这些实验而继续进行后面的结果分析。运行实验的代码如下：

```
> require(class,quietly=TRUE)
> require(randomForest,quietly=TRUE)
> require(e1071,quietly=TRUE)
> load('myALL.Rdata')
> es <- exprs(ALLb)
> # simple filtering
> ALLb <- nsFilter(ALLb,
+                  var.func=IQR,var.cutoff=IQR(as.vector(es))/5,
+                  feature.exclude="^AFFX")
> ALLb <- ALLb$eset
> # the dataset
> featureNames(ALLb) <- make.names(featureNames(ALLb))
> dt <- data.frame(t(exprs(ALLb)),Mut=ALLb$mol.bio)
> DSs <- list(dataset(Mut ~ .,dt,'ALL'))
> # The learners to evaluate
> TODO <- c('knn','svm','randomForest')
> for(td in TODO) {
+   assign(td,
+          experimentalComparison(
+               DSs,
+               c(
+                 do.call('variants',
+                         c(list('genericModel',learner=td),
+                           vars[[td]],
+                           varsRootName=td))
+                 ),
+               loocvSettings(seed=1234,verbose=F)
+                         )
+          )
+   save(list=td,file=paste(td,'Rdata',sep='.'))
+ }
```

上面的代码应用函数 experimentalComparison( ) 来测试应用 LOOCV 方法的所有变体。上面代码用函数 variants( ) 来生成变体的所有 learner 对象，从前面可知，这些变体由参数 vars 给出的列表元素提供。每一个变体由一个 LOOCV 过程来评估。上面代码的结果是 3 个 compExp 对象，它们分别命名为 knn、svm 和 randomForest。这些结果对象包含相应训练方法变体的结果，所有这些变体的结果存储在以训练方法名为文件名，以“. Rdata”为后缀的文件中。在本书的网站中有这些结果文件。因此，你如果没有运行上面的实验，你可以下载这些文件到你的本地计算机中，然后用函数 load( )（以相应文件的名称为参数）来载入 R 软件：

262

```
> load("knn.Rdata")
> load("svm.Rdata")
> load("randomForest.Rdata")
```

一个训练方法的所有变体的结果保存在相应的对象中。例如，如果需要检查哪一个支持向量机变体最好，可以用下列命令：

```
> rankSystems(svm, max = T)

$ALL
```

⊖ 在我的标准配置的桌面计算机上，运行以上代码大约需要 3 天。

虑 2 个值。

如前所述，对于每个训练方法，我们考虑 3 个不同的特征集合（由参数 fs. meth 设定）。第一个选择（list('all')）应用 ANOVA 统计检验得到的所有特征。第二个选择（list('rf', 30)）通过随机森林排序的方法，从 ANOVA 检验得到的特征中选择 30 个基因。最后一个选择应用前面描述的变量聚类组合策略选择 30 个基因，然后从变量聚类中随机选择 30 个预测变量，建立 50 个模型的组合模型。为了实现上面基于从基因聚类中选择不同的变量集的组合方法，我们建立下面的函数：

```
> varsEnsembles <- function(tgt,train,test,
+                           varsSets,
+                           baseLearner,blPars,
+                           verb=F)
+ {
+   preds <- matrix(NA,ncol=length(varsSets),nrow=NROW(test))
+   for(v in seq(along=varsSets)) {
+     if (baseLearner=='knn')
+       preds[,v] <- knn(train[,-which(colnames(train)==tgt)],
+                        test[,-which(colnames(train)==tgt)],
+                        train[,tgt],blPars)
+     else {
+       m <- do.call(baseLearner,
+                    c(list(as.formula(paste(tgt,
+                                            paste(varsSets[[v]],
+                                                  collapse='+'),
+                                            sep='~')),
+                           train[,c(tgt,varsSets[[v]])]),
+                      blPars)
+                    )
+       if (baseLearner == 'randomForest')
+         preds[,v] <- do.call('predict',
+                              list(m,test[,c(tgt,varsSets[[v]])],
+                                   type='response'))
+       else
+         preds[,v] <- do.call('predict',
+                              list(m,test[,c(tgt,varsSets[[v]])]))
+     }
+   }
+   ps <- apply(preds,1,function(x)
+               levels(factor(x))[which.max(table(factor(x)))])
+   ps <- factor(ps,
+                levels=1:nlevels(train[,tgt]),
+                labels=levels(train[,tgt]))
+   if (verb) structure(ps,ensemblePreds=preds) else ps
+ }
```

上面函数的第一部分参数是目标变量名、训练集、测试集。下一个参数（varsSets）是一个含有变量名（得到的聚类）集合的列表，我们应该从该变量集合中抽取一个变量作为组合的每一个成员的预测变量。最后两个参数（baseLearner 和 blPars）给出了组合的每个成员所应用的训练方法的函数实现的名称以及相应的参数列表。上面函数的结果是组合方法对给定测试集的预测值。这些预测是组合的成员通过投票机制产生的。组合的成员之间的差别在于它们所应用的预测变量，这些预测变量由参数 varsSets 确定。5. 3. 4 节给出了这种从变量聚类过程得到预测变量集合的方法。考虑到每一种训练方法所进行的任务是相似的，因此我们建立了一个用户定义的模型函数，它把所应用的训练方法作为参数之一。函数 genericModel( ) 实现了这种方法，代码如下：

258
~
261

```
> genericModel <- function(form,train,test,
+                          learner,
+                          fs.meth,
+                          ...)
+ {
+   cat('=')
+   tgt <- as.character(form[[2]])
+   tgtCol <- which(colnames(train)==tgt)
+
+   # Anova filtering
+   f <- Anova(train[,tgt],p=0.01)
+   ff <- filterfun(f)
+   genes <- genefilter(t(train[,-tgtCol]),ff)
+   genes <- names(genes)[genes]
+   train <- train[,c(tgt,genes)]
+   test <- test[,c(tgt,genes)]
+   tgtCol <- 1
+
+   # Specific filtering
+   if (fs.meth[[1]]=='varclus') {
+     require(Hmisc,quietly=T)
+     v <- varclus(as.matrix(train[,-tgtCol]))
+     VSs <- lapply(1:fs.meth[[3]],function(x)
+                   getVarsSet(v$hclust,nvars=fs.meth[[2]]))
+     pred <- varsEnsembles(tgt,train,test,VSs,learner,list(...))
+
+   } else {
+     if (fs.meth[[1]]=='rf') {
+       require(randomForest,quietly=T)
+       rf <- randomForest(form,train,importance=T)
+       imp <- importance(rf)
+       imp <- imp[,ncol(imp)-1]
+       rf.genes <- names(imp)[order(imp,decreasing=T)[1:fs.meth[[2]]]]
+       train <- train[,c(tgt,rf.genes)]
+       test <- test[,c(tgt,rf.genes)]
+     }
+
+     if (learner == 'knn')
+       pred <- kNN(form,
+            train,
+            test,
+            norm.stats=list(rowMedians(t(as.matrix(train[,-tgtCol]))),
+                            rowIQRs(t(as.matrix(train[,-tgtCol])))),
+            ...)
+     else {
+       model <- do.call(learner,c(list(form,train),list(...)))
+       pred <- if (learner != 'randomForest') predict(model,test)
+               else predict(model,test,type='response')
+     }
+
+   }
+
+   c(accuracy=ifelse(pred == resp(form,test),100,0))
+ }
```

在函数 LOOCV 的每一次迭代中，将调用上面的用户定义函数。在微阵列数据上完成所有这

```
$ALL$accuracy
  system    score
1 svm.v2 86.17021
2 svm.v3 86.17021
3 svm.v5 86.17021
4 svm.v6 86.17021
5 svm.v9 86.17021
```

函数 rankSystems( ) 用一个类 compExp 对象为参数，获得实验过程中所估计的每一个评估指标的最好变体。在默认情况下，该函数假设最小值意味着“最好”。如果最大值意味着“最好”，例如正确率，则可以如上所示应用参数 max [⊖]。为了得到所有实验的总体情况，可以用下面代码把 3 个结果对象结合在一起：

```
> all.trials <- join(svm, knn, randomForest, by = "variants")
```

通过结合在一起的 compExp 对象，可以检查我们实验中总体最好的分数：

```
> rankSystems(all.trials, top = 10, max = T)

$ALL
$ALL$accuracy
             system    score
1            knn.v2 88.29787
2            knn.v3 87.23404
3  randomForest.v4 87.23404
4  randomForest.v6 87.23404
5            svm.v2 86.17021
6            svm.v3 86.17021
7            svm.v5 86.17021
8            svm.v6 86.17021
9            svm.v9 86.17021
10          svm.v23 86.17021
```

获得最高分数的是 k 近邻方法的一个变体。下面查看该变体的特征：

```
> getVariant("knn.v2", all.trials)

Learner::  "genericModel"

Parameter values
         learner  =  "knn"
         k  =  5
         norm  =  TRUE
         fs.meth  =  list("rf", 30)
```

该变体应用随机森林过滤出的 30 个基因、5 个近邻，并对基因表达水平进行标准化。有趣的是，在最高的 10 个分值中，只有最后一个（“svm. v23”）的 30 个基因不是应用随机森林过滤得到的。这第 10 位最好的模型应用 ANOVA 过滤出的所有基因。注意，这 10 个模型的正确率分值很相似。事实上，考虑到我们有 94 个测试样本，最好模型的正确率意味着有 11 个样本分类错误，而第 10 位最好的模型有 13 个错误。

也许需要知道模型（例如，最好的模型）犯了哪种类型的错误。混淆矩阵提供了这种类型的信息。为了得到混淆矩阵，需要知道模型预测值对应的真实值。我们的用户定义函数没有输出

⊖ 如果我们有多个评估指标，有些指标需要最小化，其他需要最大化。可以设置参数 max 为布尔值向量。

预测值，它只给出了决策正确率。因此，compExp 对象没有这种信息。如果我们需要这种额外的信息，在实验过程的每一次迭代中，在计算评估指标的顶部，可以使用户定义函数返回这些信息并赋给实验比较过程。如第 4 章所示，这里用于接收和存储这些额外信息。设想我们需要知道最好的模型在 LOOCV 过程的每一次迭代中的预测值。下面的代码让我们获取这些信息。下面的代码着重于最好的模型，当然可以简单地修改使它可以应用于其他模型。

```
> bestknn.loocv <- function(form,train,test,...) {
+   require(Biobase,quietly=T)
+   require(randomForest,quietly=T)
+   cat('=')
+   tgt <- as.character(form[[2]])
+   tgtCol <- which(colnames(train)==tgt)
+   # Anova filtering
+   f <- Anova(train[,tgt],p=0.01)
+   ff <- filterfun(f)
+   genes <- genefilter(t(train[,-tgtCol]),ff)
+   genes <- names(genes)[genes]
+   train <- train[,c(tgt,genes)]
+   test <- test[,c(tgt,genes)]
+   tgtCol <- 1
+   # Random Forest filtering
+   rf <- randomForest(form,train,importance=T)
+   imp <- importance(rf)
+   imp <- imp[,ncol(imp)-1]
+   rf.genes <- names(imp)[order(imp,decreasing=T)[1:30]]
+   train <- train[,c(tgt,rf.genes)]
+   test <- test[,c(tgt,rf.genes)]
+   # knn prediction
+   ps <- kNN(form,train,test,norm=T,
+             norm.stats=list(rowMedians(t(as.matrix(train[,-tgtCol]))),
+                             rowIQRs(t(as.matrix(train[,-tgtCol])))),
+             k=5,...)
+   structure(c(accuracy=ifelse(ps == resp(form,test),100,0)),
+             itInfo=list(ps)
+            )
+ }
> resTop <- loocv(learner('bestknn.loocv',pars=list()),
+                 dataset(Mut~.,dt),
+                 loocvSettings(seed=1234,verbose=F),
+                 itsInfo=T)
```

函数 bestknn. loocv( ) 实际上是前面提及的函数 genericModel( ) 的特例，它设定 5 个近邻，应用随机森林过滤法，应用中位数和四分位距来进行标准化。除了它的返回结果部分以外，大部分的代码和函数 genericModel( ) 一样。这个新函数不是返回模型的正确率，它返回一个结构。如之前描述的那样，结构是一种有附加属性的 R 对象。函数 structure( ) 可以通过附加一组属性来创建这种“丰富”的对象。在我们的用户定义函数中，如果我们需要为函数 loocv( ) 返回一些额外信息，我们就应该在一个名为 itsInfo 的属性中完成。在上面的函数中，我们应用这个属性返回模型的预测值。在函数 loocv( ) 中存储实验过程的每一次迭代给出的这个信息。为了让函数 loocv( ) 保存这些信息，需要应用可选参数 itsInfo = T 来调用函数 loocv( )。这可以确保用户自定义函数返回的任何内容可以作为一个名为 itsInfo 的属性，该属性将收集在一个列表中，并作为函数 loocv( ) 的结果中名为 itsInfo 的属性返回。最后，我们可以检查这一信息，即检查每一次迭代

263
~
265

中最好模型的预测值。

为了检查包含我们所需要信息的属性内容，可以应用如下代码（下面只显示了开始的4个预测值）：

```
> attr(resTop, "itsInfo")[1:4]

[[1]]
[1] BCR/ABL
Levels: ALL1/AF4 BCR/ABL E2A/PBX1 NEG

[[2]]
[1] NEG
Levels: ALL1/AF4 BCR/ABL E2A/PBX1 NEG

[[3]]
[1] BCR/ABL
Levels: ALL1/AF4 BCR/ABL E2A/PBX1 NEG

[[4]]
[1] ALL1/AF4
Levels: ALL1/AF4 BCR/ABL E2A/PBX1 NEG
```

应用函数 attr( ) 可以获取一个 R 对象的任何属性值。我们知道，属性 itsInfo 含有 LOOCV 过程每次迭代的预测值。应用这些信息和数据集的真实值，可以得到混淆矩阵，代码如下：

```
> preds <- unlist(attr(resTop, "itsInfo"))
> table(preds, dt$Mut)

preds      ALL1/AF4 BCR/ABL E2A/PBX1 NEG
  ALL1/AF4       10       0        0   0
  BCR/ABL         0      33        0   4
  E2A/PBX1        0       0        3   1
  NEG             0       4        2  37
```

可以用混淆矩阵来检查模型所犯错误的类型。例如，模型正确地预测了所有的突变类型为 ALL1/AF4 的样本。同时，我们观测到模型所犯的大部分错误是把一个具有某些突变的个案预测为类 NEG。然而，模型也把5个没有突变的样本错误地预测为有某些异常。 266

5.5 小结

本章的主要目的是介绍 R 社区十分关注的数据挖掘领域：生物信息学中的一系列应用。这里探索了 Bioconductor 项目的一些工具，该项目提供了专用于生物信息学中应用问题的大量 R 添加包。作为一个具体的例子，我们着重于生物信息学预测任务：预测与患有急性淋巴细胞白血病的病人样本相关的基因突变类型。基于从微阵列实验得到的一组基因的表达水平信息，建立了多个分类模型。就数据挖掘概念而言，本章着重于下列主题：

- 有大量预测变量问题的特征选择方法。
- 分类方法。
- 随机森林。
- *k* 近邻方法。
- 支持向量机。
- 用不同的预测变量集合进行组合。

- 弃一交叉验证法。

对于 R 而言，我们学习了几种新的技术：

- 如何处理微阵列数据。
- 应用 ANOVA 检验比较几组数据的均值。
- 在分类问题中用随机森林来选择变量。
- 变量聚类。
- 获得用不同预测变量建立的模型的组合。
- 获得 k 近邻模型。
- 用弃一交叉验证法估计模型的精度。

267



1. Acuna, E., and Members of the CASTLE group at UPR-Mayaguez, (2009). *dprep: Data preprocessing and visualization functions for classification.* R package version 2.1.

2. Adler, D., Glaser, C., Nenadic, O., Oehlschlagel, J., and Zucchini, W. (2010). *ff: Memory-efficient storage of large data on disk and fast access functions.* R package version 2.1-2.

3. Aha, D. (1990). *A Study of Instance-Based Learning Algorithms for Supervised Learning Tasks: Mathematical, Empirical, and Psychological Evaluations.* Ph.D. thesis, University of California at Irvine, Department of Information and Computer Science.

4. Aha, D. (1997). Lazy learning. *Artificial Intelligence Review*, 11, 7–10.

5. Aha, D., Kibler, D., and Albert, M. (1991). Instance-based learning algorithms. *Machine Learning*, 6(1):37–66.

6. Atkeson, C. G., Moore, A., and Schaal, S. (1997). Locally weighted learning. *Artificial Intelligence Review*, 11:11–73.

7. Austin, J. and Hodge, V. (2004). A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22:85–126.

8. Barnett, V. and Lewis, T. (1994). *Outliers in statistical data (3rd edition).* John Wiley.

9. Bontempi, G., Birattari, M., and Bersini, H. (1999). Lazy learners at work: The lazy learning toolbox. In *Proceedings of the 7th European Congress on Intelligent Tecnhiques & Soft Computing (EUFIT'99).*

10. Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24:123–140.

11. Breiman, L. (1998). Arcing classifiers (with discussion). *Annals of Statistics*, 26:801–849.

12. Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

13. Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). *Classification and regression trees.* Statistics/Probability Series. Wadsworth & Brooks/Cole Advanced Books & Software.

14. Breunig, M., Kriegel, H., Ng, R., and Sander, J. (2000). LOF: identifying density-based local outliers. In *ACM Int. Conf. on Management of Data*, pages 93–104.

15. Carl, P. and Peterson, B. G. (2009). *PerformanceAnalytics: Econometric tools for performance and risk analysis.* R package version 1.0.0.

16. Chambers, J. (2008). *Software for data analysis: Programming with R.* Springer.

17. Chan, R. (1999). Protecting rivers & streams by monitoring chemical concentrations and algae communities. In *Proceedings of the 7th European Congress on Intelligent Tecnhiques & Soft Computing (EUFIT'99).*

18. Chandola, V., Banerjee, A., and Kumar, V. (2007). Outlier detection: A survey. Technical Report TR 07-017, Department of Computer Science and Engineering, University of Minnesota.

19. Chatfield, C. (1983). *Statistics for technology.* Chapman and Hall, 3rd edition.

20. Chawla, N. (2005). *The data mining and knowledge discovery handbook*, chapter on data mining for imbalanced datasets: an overview, pages 853–867. Springer.

21. Chawla, N., Japkowicz, N., and Kokz, A. (2004). SIGKDD *Explorations* special issue on learning from imbalanced datasets.

22. Chawla, N., Lazarevic, A., Hall, L., and Bowyer, K. (2003). Smote-boost: Improving prediction of the minority class in boosting. In *Seventh European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 107–119.

23. Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357.

24. Chen, C., Hardle, W., and Unwin, A., Editors (2008). *Handbook of data visualization.* Springer.

25. Chiaretti, S., Li, X., Gentleman, R., Vitale, A., Vignetti, M., Mandelli, F., Ritz, J., and Foa, R. (2004). Gene expression profile of adult T-cell acute lymphocytic leukemia identifies distinct subsets of patients with different response to therapy and survival. *Blood*, 103(7), 2771–2778.

26. Chizi, B. and Maimon, O. (2005). *The data mining and knowledge discovery handbook*, chapter on dimension reduction and feature selection, pages 93–111. Springer.

27. Cleveland, W. (1993). *Visualizing data.* Hobart Press.

28. Cleveland, W. (1995). *The elements of graphing data.* Hobart Press.

29. Cleveland, W. and Loader, C. (1996). Smoothing by local regression: Principles and methods, statistical theory and computational aspects of smoothing. Edited by W. Haerdle and M. G. Schimek, Springer, 10–49.

30. Cortes, E. A., Martinez, M. G., and Rubio, N. G. (2010). *adabag: Applies Adaboost.M1 and Bagging.* R package version 1.1.

31. Cover, T. M. and Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.

32. Dalgaard, P. (2002). *Introductory statistics with R.* Springer.

33. Deboeck, G., Editor. (1994). *Trading on the edge.* John Wiley & Sons.

34. Demsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30.

35. Devogelaere, D., Rijckaert, M., and Embrechts, M. J. (1999). 3rd international competition: Protecting rivers and streams by monitoring chemical concentrations and algae communities solved with the use of gadc. In *Proceedings of the 7th European Congress on Intelligent Tecnhiques & Soft Computing (EUFIT'99).*

36. Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10:1895–1923.

37. Dietterich, T. G. (2000). Ensemble methods in machine learning. *Lecture Notes in Computer Science*, 1857:1–15.

38. Dimitriadou, E., Hornik, K., Leisch, F., Meyer, D., and Weingessel, A. (2009). *e1071: Misc Functions of the Department of Statistics (e1071), TU Wien.* R package version 1.5-19.

39. Domingos, P. (1999). Metacost: A general method for making classifiers cost-sensitive. In *KDD'99: Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining*, pages 155–164. ACM Press.

40. Domingos, P. and Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29:103–137.

41. Drapper, N. and Smith, H. (1981). *Applied Regression Analysis.* John Wiley & Sons, 2nd edition.

42. Drummond, C. and Holte, R. (2006). Cost curves: An improved method for visualizing classifier performance. *Machine Learning*, 65(1):95–130.

43. DuBois, P. (2000). *MySQL.* New Riders.

44. Elkan, C. (2001). The foundations of cost-sensitive learning. In *IJCAI'01: Proceedings of 17th International Joint Conference of Artificial Intelligence*, pages 973–978. Morgan Kaufmann Publishers Inc.

45. Fox, J. (2009). *car: Companion to Applied Regression.* R package version 1.2-16.

46. Freund, Y. (1990). Boosting a weak learning algorithm by majority. In *Proceedings of the Third Annual Workshop on Computational Learning Theory.*

47. Freund, Y. and Shapire, R. (1996). Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning.* Morgan Kaufmann.

48. Friedman, J. (1991). Multivariate adaptive regression splines. *The Annals of Statistics*, 19(1):1–144.

49. Friedman, J. (2002). Stochastic gradient boosting. *Computational Statistics and Data Analysis*, 38(4):367–378.

50. Gama, J. and Gaber, M., Editors. (2007). *Learning from data streams.* Springer.

51. Gama, J., Medas, P., Castillo, G., and Rodrigues, P. (2004). Learning with drift detection. In Bazzan, A. and Labidi, S., Editors, *Advances in artificial intelligence-SBIA 2004*, volume 3171 of *Lecture Notes in Computer Science*, pages 286–295. Springer.

52. Gentleman, R., Carey, V., Huber, W., and Hahne, F. (2010). *genefilter: genefilter: methods for filtering genes from microarray experiments.* R package version 1.28.2.

53. Gentleman, R. C., Carey, V. J., Bates, D. M., et al. (2004). Bioconductor: Open software development for computational biology and bioinformatics. *Genome Biology*, 5:R80.

54. Hahne, F., Huber, W., Gentleman, R., and Falcon, S. (2008). *Bioconductor case studies.* Springer.

55. Han, J. and Kamber, M. (2006). *Data mining: concepts and techniques (2nd edition).* Morgan Kaufmann Publishers.

56. Hand, D., Mannila, H., and Smyth, P. (2001). *Principles of data mining.* MIT Press.

57. Hand, D. and Yu, K. (2001). Idiot's Bayes — Not so stupid after all? *International Statistical Review*, 69(3):385–399.

58. Hand, D. J. (2009). Measuring classifier performance: A coherent alternative to the area under the roc curve. *Machine Learning*, 77(1):103–123.

59. Harrell, Jr., F. E. (2009). *Hmisc: Harrell miscellaneous.* R package version 3.7-0. With contributions from many other users.

60. Hastie, T. and Tibshirani, R. (1990). *Generalized additive models.* Chapman & Hall.

61. Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The elements of statistical learning: data mining, inference and prediction.* Springer.

62. Hawkins, D. M. (1980). *Identification of outliers.* Chapman and Hall.

63. Hong, S. (1997). Use of contextual information for feature ranking and discretization. *IEEE Transactions on Knowledge and Data Engineering*, 9(5):718–730.

64. Hornik, K., Buchta, C., and Zeileis, A. (2009). Open-source machine learning: R meets Weka. *Computational Statistics*, 24(2):225–232.

65. Ihaka, R. and Gentleman, R. (1996). R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314.

66. James, D. A. and DebRoy, S. (2009). *RMySQL: R interface to the MySQL database.* R package version 0.7-4.

67. Japkowicz, N. (2000). The class imbalance problem: Significance and strategies. In *Proceedings of the 2000 International Conference on Artificial Intelligence (IC-A1'2000):Special Track on Inductive Learning.*

68. Karatzoglou, A., Smola, A., Hornik, K., and Zeileis, A. (2004). kernlab — an S4 package for kernel methods in R. *Journal of Statistical Software*, 11(9):1–20.

69. Kaufman, L. and Rousseeuw, P. (1990). *Finding groups in data: An introduction to cluster analysis.* John Wiley & Sons, New York.

70. Kifer, D., Ben-David, S., and Gehrke, J. (2004). Detecting change in data streams. In *VLDB 04: Proceedings of the 30th International Conference on Very Large Data Bases*, pages 180–191. Morgan Kaufmann.

71. Kira, K. and Rendel, L. (1992). The feature selection problem: Traditional methods and a new algorithm. In *Proc. Tenth National Conference on Artificial Intelligence*, pages 129–134. MIT Press.

72. Klinkenberg, R. (2004). Learning drifting concepts: example selection vs. example weighting. *Intelligent Data Analysis*, 8(3):281–300.

73. Kononenko, I. (1991). Semi-naive Bayesian classifier. In *EWSL-91: Proceedings of the European Working Session on Learning on Machine Learning*, pages 206–219. Springer.

74. Kononenko, I., Simec, E., and Robnik-Sikonja, M. (1997). Overcoming the myopia of induction learning algorithms with relieff. *Applied Intelligence*, 17(1):39–55.

75. Kubat, M. and Matwin, S. (1997). Addressing the curse of imbalanced training sets: one-sided selection. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 179–186.

76. Leisch, F., Hornik, K., and Ripley., B. D. (2009). *mda: Mixture and flexible discriminant analysis, S original by Trevor Hastie and Robert Tibshirani.* R package version 0.3-4.

77. Li, X. (2009). *ALL: A data package.* R package version 1.4.7.

78. Liaw, A. and Wiener, M. (2002). Classification and regression by randomforest. *R News*, 2(3):18–22.

79. Liu, H. and Motoda, H. (1998). *Feature selection for knowledge discovery and data mining.* Kluwer Academic Publishers.

80. McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133.

81. Milborrow, S. (2009). *Earth: Multivariate adaptive regression spline models, derived from mda:mars by Trevor Hastie and Rob Tibshirani.* R package version 2.4-0.

82. Minsky, M. and Papert, S. (1969). *Perceptrons: an introduction to computational geometry.* MIT Press.

83. Murrell, P. (2006). *R graphics.* Chapman & Hall/CRC.

84. Murtagh, F. (1985). Multidimensional clustering algorithms. *COMPSTAT Lectures 4*, Wuerzburg: Physica-Verlag.

85. Myers, R. (1990). *Classical and modern regression with applications.* 2nd edition. Duxbury Press.

86. Nadaraya, E. (1964). On estimating regression. *Theory of Probability and its Applications*, 9:141–142.

87. Nemenyi, P. (1969). Distribution-free Multiple Comparisons. Ph.D. thesis, Princeton University.

88. Ng, R. and Han, J. (1994). Efficient and effective clustering method for spatial data mining. In *Proceedings of the 20th International Conference on Very Large Data Bases*, page 144. Morgan Kaufmann.

89. Oakland, J. (2007). *Statistical process control, 6th edition.* Butterworth-Heinemann.

90. Provost, F. and Fawcett, T. (1997). Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In *KDD'97: Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, pages 43–48. AAAI Press.

91. Provost, F. and Fawcett, T. (2001). Robust classification for imprecise environments. *Machine Learning*, 42(3), 203–231.

92. Provost, F., Fawcett, T., and Kohavi, R. (1998). The case against accuracy estimation for comparing induction algorithms. In *Proc. 15th International Conf. on Machine Learning*, pages 445–453. Morgan Kaufmann, San Francisco, CA.

93. Pyle, D. (1999). *Data preparation for data mining.* Morgan Kaufmann.

94. Quinlan, R. (1993). *C4.5: programs for machine learning.* Morgan Kaufmann.

95. R Special Interest Group on Databases, R.-S.-D. (2009). *DBI: R Database Interface.* R package version 0.2-5.

96. Rätsch, G., Onoda, T., and Müller, K. (2001). Soft margins for *AdaBoost*. *Machine Learning*, 42(3):287–320.

97. Rijsbergen, C. V. (1979). *Information retrieval.* 2nd edition. Dept. of Computer Science, University of Glasgow.

98. Rish, I. (2001). An empirical study of the Naive Bayes classifier. In *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, pages 41–46.

99. Rogers, R. and Vemuri, V. (1994). *Artificial neural networks forecasting time series.* IEEE Computer Society Press.

100. Rojas, R. (1996). *Neural networks.* Springer-Verlag.

101. Ronsenblatt, F. (1958). The perceptron: A probabilistic models for information storage and organization in the brain. *Psychological Review*, 65:386–408.

102. Rosenberg, C., Hebert, M., and Schneiderman, H. (2005). Semi-supervised self-training of object detection models. In *Proceedings of the 7th IEEE Workshop on Applications of Computer Vision*, pages 29–36. IEEE Computer Society.

103. Rumelhart, D., Hinton, G., and Williams, R. (1986). Learning internal representations by error propagation. In Rumelhart, D.E. et al., Editors, *Parallel distributed processing*, volume 1. MIT Press.

104. Ryan, J. A. (2009). *quantmod: Quantitative financial modelling framework.* R package version 0.3-13.

105. Ryan, J. A. and Ulrich, J. M. (2010). *xts: Extensible time series.* R package version 0.7-0.

106. Sarkar, D. (2010). *lattice: Lattice graphics.* R package version 0.18-3.

107. Seeger, M. (2002). Learning with Labeled and Unlabeled Data. Technical report, Institute for Adaptive and Neural Computation, University of Edinburgh.

108. Shapire, R. (1990). The strength of weak learnability. *Machine Learning*, 5:197–227.

109. Shawe-Taylor, J. and Cristianini, N. (2000). *An introduction to support vector machines.* Cambridge University Press.

110. Sing, T., Sander, O., Beerenwinkel, N., and Lengauer, T. (2009). *ROCR: Visualizing the performance of scoring classifiers.* R package version 1.0-4.

111. Smola, A. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, 14:199–222.

112. Smola, A. J. and Schölkopf, B. (1998). A tutorial on support vector regression. NeuroCOLT Technical Report TR-98-030.

113. Therneau, T. M. and Atkinson, B.; port by Brian Ripley. (2010). *rpart: Recursive Partitioning.* R package version 3.1-46.

114. Torgo, L. (1999a). Inductive Learning of Tree-based Regression Models. Ph.D. thesis, Faculty of Sciences, University of Porto.

115. Torgo, L. (1999b). Predicting the density of algae communities using local regression trees. In *Proceedings of the 7th European Congress on Intelligent Tecnhiques & Soft Computing (EUFIT'99).*

116. Torgo, L. (2000). Partial linear trees. In Langley, P., Editor, *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pages 1007–1014. Morgan Kaufmann.

117. Torgo, L. (2007). Resource-bounded fraud detection. In Neves, J. et. al, Editors, *Proceedings of the 13th Portuguese Conference on Artificial Intelligence (EPIA'07)*, pages 449–460, Springer.

118. Trapletti, A. and Hornik, K. (2009). *tseries: Time series analysis and computational finance.* R package version 0.10-22.

119. Ulrich, J. (2009). *TTR: Technical trading rules.* R package version 0.20-1.

120. Vapnik, V. (1995). *The nature of statistical learning theory.* Springer.

121. Vapnik, V. (1998). *Statistical Learning Theory.* John Wiley & Sons.

122. Venables, W. N. and Ripley, B. D. (2002). *Modern applied statistics with S.* fourth edition, Springer.

123. Watson, G. S. (1964). Smooth regression analysis. *Sankhya: The Indian Journal of Statistics, Series A*, 26:359–372.

124. Weihs, C., Ligges, U., Luebke, K., and Raabe, N. (2005). klar analyzing German business cycles. In Baier, D., Decker, R., and Schmidt-Thieme, L., Editors, *Data analysis and decision support*, pages 335–343, Springer-Verlag.

125. Weiss, G. and F. Provost (2003). Learning when training data are costly: The effect of class distribution on tree induction. *Journal of Artificial Intelligence Research*, 19:315–354.

126. Weiss, S. and Indurkhya, N. (1999). *Predictive data mining.* Morgan Kaufmann.

127. Werbos, P. (1974). *Beyond Regression — New Tools for Prediction and Analysis in the Behavioral Sciences.* Ph.D. thesis, Harvard University.

128. Werbos, P. (1996). *The roots of backpropagation — from observed derivatives to neural networks and political forecasting.* John Wiley & Sons.

129. Wettschereck, D. (1994). *A Study of Distance-Based Machine Learning Algorithms.* Ph.D. thesis, Oregon State University.

130. Wettschereck, D., Aha, D., and Mohri, T. (1997). A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, 11:11–73.

131. Wilson, D. and Martinez, T. (1997). Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6:1–34.

132. Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 189–196.

133. Zeileis, A. and Grothendieck, G. (2005). zoo: S3 infrastructure for regular and irregular time series. *Journal of Statistical Software*, 14(6):1–27.

134. Zhu, X. (2005). Semi-Supervised Learning with Graphs. Ph.D. thesis, School of Computer Science, Carnegie Mellon University.

135. Zhu, X. (2006). Semi-Supervised Literature Survey. Technical report TR 1530, University of Wisconsin–Madison.

136. Zirilli, J. (1997). *Financial prediction using neural networks.* International Thomson Computer Press.

索引中的页码为英文原书页码，与书中页边标注的页码一致。

M

N

O

P

R

数据挖掘术语索引

索引中的页码为英文原书页码，与书中页边标注的页码一致。

R 函数索引

索引中的页码为英文原书页码，与书中页边标注的页码一致。

